NETCONF	B. Lengyel
Internet-Draft	Ericsson
Intended status: Standards Track	M. Bjorklund
Expires: May 4, 2009	Tail-f Systems
	October 31, 2008

Partial Lock RPC for NETCONF
draft-ietf-netconf-partial-lock-04

Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with Section 6 of BCP 79.
Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.
Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as “work in progress.”
The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.
The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.
This Internet-Draft will expire on May 4, 2009.

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 [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.

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

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 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 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.

2.2. Dependencies

The device MUST support restricted XPath expressions in the select element, as described in Section 2.4.1 (<partial-lock>). Optionally, if the :xpath capability is also supported, 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 element in the <partial-lock> operation. Each XPath expression MUST return a node set. Locking a node also protects the complete subtree under the node from modification by others.

In some situations it is desirable that the same set of nodes are locked in more than one datastore. As an example: if an interface is configured in the candidate datastore, it is dangerous if it is 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 one or multiple datastores.

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 datastore(s) or none. That is, if during the <partial-lock> operation one of the requested parts cannot be locked, the server MUST unlock all parts that were previously 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 no longer covered by the lock. If all top level nodes are deleted, the lock will still be present. However, its scope will become nil (i.e., 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 scope of the individual locks overlaps, nodes in the common area MUST be locked 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 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 rights to all parts of a 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.

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 that operation fails, the client SHOULD back down, release any previously acquired locks, and retry the procedure after waiting some time interval. The length of the interval SHOULD be random to avoid repeated dead-locks when both (or all) clients back down and then retry 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. There are some other issues that are intentionally not addressed:

* 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 in the unlocked part of the datastore), these values SHOULD be included in the scope of the lock.

- *Configuration data can be edited both inside and outside the scope 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] (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 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

TOC

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 used for all of them. For each datastore locking of the same set of nodes will be requested.
>
> **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 that matched the select expression and are locked are reported in the rpc-reply message. Note that if some of the requested nodes exist only in one of the 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 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 select expression returns something other than 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

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 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>

The operation unlocks the parts of the datastores that were 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 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

TOC

A successful partial lock will cause a subsequent operation to fail if it attempts to modify the locked area and is executed in a NETCONF session other than the session that owns the lock. All operations that modify the datastore are affected, including: <edit-config>, <copy-config>, <delete-config>, <commit> and <discard-changes>. A successful partial lock will also cause the (global) <lock> operation to fail. All of these operations are affected only if they are targeting the same datastore.

2.6. Interactions with Other Capabilities

TOC

2.6.1. Candidate Configuration Capability

TOC

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

TOC

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> 'Lock denied, Outstanding confirmed commit' is returned.

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.

3. Security Considerations

TOC

The same considerations are relevant as for the base NETCONF Protocol [NETCONF] (Enns, R., “NETCONF Configuration Protocol,” December 2006.). 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

TOC

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
<?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="urn:ietf:params:xml:ns:netconf:base:1.0"/>

  <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">
            <xs:annotation>
              <xs:documentation>
                A list of one or more datastore names for NETCONF.
              </xs:documentation>
            </xs:annotation>
            <xs:complexType>
              <xs:sequence>
                <xs:element name="startup" minOccurs="0">
                  <xs:complexType/>
                </xs:element>
                <xs:element name="candidate" minOccurs="0">
                  <xs:complexType/>
                </xs:element>
                <xs:element name="running" minOccurs="0">
                  <xs:complexType/>
                </xs:element>
              </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="xs:unsignedInt">
          <xs:annotation>
            <xs:documentation>
              Identifies the lock, SHOULD be used in the subsequent
              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="dataPartInPpartialLockReplyType">
  <xs:annotation>
    <xs:documentation>
      In a reply to a successful partial-lock request the content
      of the <rpc-reply> element MUST conform to this complex type.
    </xs:documentation>
  </xs:annotation>
  <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: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] (Bjorklund, M., “YANG - A data modeling language for NETCONF,” August 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
     balazs.lengyel@ericsson.com";

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

  revision 2008-10-31 {
    description "Initial version.";
  }

  grouping configNames {
    container target {
        description
            "A list of one or more datastore names.";
        leaf running   { type empty; }
        leaf candidate { type empty; }
        leaf startup   { type empty; }
        must "running or candidate or startup" {
            error-message "At least one datastore must be specified.";
        }
     }
  }

  rpc partial-lock {
    description
      "A NETCONF operation that locks part of one or more datastores.";
    input {
      uses configNames;
      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;
      }
      container running {
          leaf-list locked-node {
              description "List of locked nodes
                           in the running datastore";
              type instance-identifier;
          }
      }
      container candidate {
          leaf-list locked-node {
              description "List of locked nodes
                           in the candidate datastore";
              type instance-identifier;
          }
      }
      container startup {
          leaf-list locked-node {
              description "List of locked nodes
                           in the startup datastore";
              type instance-identifier;
          }
      }
    }
  }

  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. 03-04

TOC

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.

7.2. 02-03

TOC

Minor clarifications
Same descriptions in XSD and YANG.

7.3. 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.

7.4. 00-01

TOC

Added YANG module.

TOC

7.5. -00

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

8. Acknowledgements

TOC

Thanks to Andy Bierman, Sharon Chisholm, Phil Shafer , David Harrington, Mehmet Ersue, Wes Hardaker 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-01 (work in progress), August 2008 (TXT).

Authors' Addresses

TOC

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

Full Copyright Statement

TOC

Copyright © The IETF Trust (2008).

This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an “AS IS” basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY, THE IETF TRUST AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.