Workgroup: NETCONF
Internet-Draft:
draft-lindblad-netconf-transaction-id-01
Published: 22 October 2021
Intended Status: Standards Track
Expires: 25 April 2022
Authors: J. Lindblad
Cisco Systems

Transaction ID Mechanism for NETCONF

Abstract

NETCONF clients and servers often need to have a synchronized view of the server's configuration data stores. The volume of configuration data in a server may be very large, while data store changes typically are small when observed at typical client resynchronization intervals.

Rereading the entire data store and analyzing the response for changes is an inefficient mechanism for synchronization. This document specifies an extension to NETCONF that allows clients and servers to keep synchronized with a much smaller data exchange and without any need for servers to store information about the clients.

Discussion Venues

This note is to be removed before publishing as an RFC.

Source for this draft and an issue tracker can be found at https://github.com/janlindblad/netconf-transaction-id.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

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."

This Internet-Draft will expire on 25 April 2022.

Copyright Notice

Table of Contents

1. Introduction

When a NETCONF client connects with a NETCONF server, a frequently occurring use case is for the client to find out if the configuration has changed since it was last connected. Such changes could occur for example if another NETCONF client has made changes, or another system or operator made changes through other means than NETCONF.

One way of detecting a change for a client would be to retrieve the entire configuration from the server, then compare the result with a previously stored copy at the client side. This approach is not popular with most NETCONF users, however, since it would often be very expensive in terms of communications and computation cost.

Furthermore, even if the configuration is reported to be unchanged, that will not guarantee that the configuration remains unchanged when a client sends a subsequent change request, a few moments later.

Evidence of a transaction id feature being demanded by clients is that several server implementors have built proprietary and mutually incompatible mechanisms for obtaining a transaction id from a NETCONF server.

RESTCONF, RFC 8040, defines a mechanism for detecting changes in configuration subtrees based on Entity-tags (ETags). In conjunction with this, RESTCONF provides a way to make configuration changes conditional on the server confiuguration being untouched by others. This mechanism leverages RFC 7232 "Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests".

This document defines similar functionality for NETCONF, RFC 6241.

2. Conventions and Definitions

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] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. NETCONF Transaction id Extension

This document describes a NETCONF extension which modifies the behavior of get-config, get-data, edit-config, edit-data, discard-changes, copy-config, delete-config and commit such that clients are able to conditionally retrieve and update the configuration in a NETCONF server. NETCONF servers that support this extension MUST announce the capability "urn:ietf:params:netconf:capability:txid:1.0".

Several low level mechanisms could be defined to fulfill the requirements for efficient client-server transaction id synchronization. This document defines only one mechanism, but additional mechanisms could be added in future versions of this document, or in separate documents.

The common use cases for such mecahnisms are briefly discussed here.

Initial configuration retrieval
When the client initially connects to a server, it may be interested to acquire a current view of (parts of) the server's configuration.
In order to be able to efficiently detect changes later, it may also be interested to store meta level transaction id information about subtrees of the configuration.

Subsequent configuration retrieval When a client needs to reread (parts of) the server's configuration, it may be interested to leverage the transaction id meta data it has stored by requesting the server to prune the response so that it does not repeat configuration data that the client is already aware of.

Configuration update with transaction id return When a client issues a transaction towards a server, it may be interested to also learn the new transaction id meta data the server has stored for the updated parts of the configuration.

Configuration update with transaction id specification When a client issues a transaction towards a server, it may be interested to also specify the new transaction id meta data that the server stores for the updated parts of the configuration.

Conditional configuration update When a client issues a transaction towards a server, it may specify transaction id data for the transaction in order to allow the server to verify that the client is up to date with any changes in the parts of the configuration that it is concerned with. If the transaction id information in the server is different than the client expected, the server rejects the transaction with a specific error message.

3.1. General Principles

All transaction id mechanisms SHALL maintain a transaction id value for each configuration datastore supported by the server. Some transaction id mechanisms will also maintain transaction id values for elements deeper in the YANG data tree. The elements for which the server maintains transaction ids are collectively referred to as the "versioned elements".

The server returning transaction id values for the versioned elements MUST ensure the transaction id values are changed every time there has been a configuration change at or below the element associated with the value. This means any update of a config true element will result in a new transaction id value for all ancestor versioned elements, up to and including the datastore root itself.

This also means a server MUST update the transaction id value for any elements that change as a result of a configuration change,

regardless of source, even if the changed elements are not explicitly part of the change payload. An example of this is dependent data under YANG RFC 7950 when- or choice-statements.

The server MUST NOT change the transaction id value of a versioned element unless a child element of that element has been changed. The server MUST NOT change any transaction id values due to changes in config false data.

3.2. Conditional Transactions

Conditional transactions are useful when a client is interested to make a configuration change, being sure that the server configuration has not changed since the client last inspected it.

By supplying the latest transaction id values known to the client in its change requests (edit-config etc.), it can request the server to reject the transaction in case any relevant changes have occurred at the server that the client is not yet aware of.

This allows a client to reliably compute and send confiuguration changes to a server without either acquiring a global datastore lock for a potentially extended period of time, or risk that a change from another client disrupts the intent in the time window between a read (get-config etc.) and write (edit-config etc.) operation.

If the server rejects the transaction because the configuration transaction id value differs from the client's expectation, the server MUST return an rpc-error with the following values:

```
error-tag:      operation-failed
error-type:     protocol
error-severity: error
```

Additionally, the error-info tag SHOULD contain an sx:structure containing relevant details about the mismatching transaction ids.

3.3. Other NETCONF Operations

discard-changes The discard-changes operation resets the candidate datastore to the contents of the running datastore. The server MUST ensure the transaction id values in the candidate datastore

get the same values as in the running datastore when this operation runs.

copy-config The copy-config operation can be used to copy contents between datastores. The server MUST ensure the transaction id values retain the same values as in the soruce datastore.

If copy-config is used to copy from a file, URL or other source that is not a datastore, the server MUST ensure the transaction id values are changed.

delete-config The server MUST ensure the datastore transaction id value is changed.

commit At commit, with regards to the transaction id values, the server MUST treat the contents of the candidate datastore as if any transaction id value provided by the client when updating the candidate was provided in a single edit-config towards the running datastore. If the transaction is rejected due to transaction id value mismatch, an rpc-error as described in section Conditional Transactions (Section 3.2) MUST be sent.

4. ETag Transaction id Mechanism

4.1. ETag attribute

Central to the ETag configuration retrieval and update mechanism described in the following sections is a meta data XML attribute called "etag". The etag attribute is defined in the namespace "urn:ietf:params:xml:ns:netconf:txid:1.0".

Servers MUST maintain a top-level etag value for each configuration datastore they implement. Servers SHOULD maintain etag values for YANG containers that hold configuration for different subsystems. Servers MAY maintain etag values for any YANG container or list element they implement.

The etag attribute values are opaque UTF-8 strings chosen freely, except that the etag string must not contain space, backslash or double quotes. The point of this restriction is to make it easy to reuse implementations that adhere to section 2.3.1 in RFC 7232. The probability SHOULD be made very low that an etag value that has been used historically by a server is used again by that server.

The detailed rules for when to update the etag value are described in section Configuration Update (Section 4.3). These rules are chosen to be consistent with the ETag mechanism in RESTCONF, RFC 8040, specifically sections 3.4.1.2, 3.4.1.3 and 3.5.2.

4.2. Configuration Retreival

Clients MAY request the server to return etag attribute values in the response by adding one or more etag attributes in get-config or get-data requests.

The etag attribute may be added directly on the get-config or get-data requests, in which case it pertains to the entire datastore. A client MAY also add etag attributes to zero or more individual elements in the get-config or get-data filter, in which case it pertains to the subtree rooted at that element.

For each element that the client requests etag attributes, the server MUST return etags for all versioned elements at or below that point that are part of the server's respone. ETags are returned as attributes on the element they pertain to. The datastore root etag value is returned on the top-level data tag in the response.

If the client is requesting an etag value for an element that is not among the server's versioned elements, then the server MUST return the etag attribute on the closest ancestor that is a versioned element, and all children of that ancestor. The datastore root is always a versioned element.

4.2.1. Initial Configuration Response

When the client adds etag attributes to a get-config or get-data request, it should specify the last known etag values it has seen for the elements it is asking about. Initially, the client will not know any etag value and should use "?".

To retrieve etag attributes across the entire NETCONF server configuration, a client might send:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1"
     xmlns:txid="urn:ietf:params:xml:ns:netconf:txid:1.0">
  <get-config txid:etag="?"/>
</rpc>
```

To retrieve etag attributes for a specific interface using an xpath filter, a client might send:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1"
     xmlns:txid="urn:ietf:params:xml:ns:netconf:txid:1.0">
  <get-config>
    <source>
      <running/>
    </source>
    <filter type="xpath"
      xmlns:if="urn:ietf:params:xml:ns:yang:ietf-interfaces"
      select=
        "/if:interfaces/if:interface[if:name='GigabitEthernet-0/0']"
      txid:etag="?"/>
  </get-config>
</rpc>
```

To retrieve etag attributes for "ietf-interfaces", but not for "nacm", a client might send:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1"
     xmlns:txid="urn:ietf:params:xml:ns:netconf:txid:1.0">
  <get-config>
    <source>
      <running/>
    </source>
    <filter>
      <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
        txid:etag="?"/>
      <nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm"/>
    </filter>
  </get-config>
</rpc>
```

When a NETCONF server receives a get-config or get-data request containing txid:etag attributes with the value "?", it MUST return etag attributes for all versioned elements below this point included in the reply.

If the server considers the container "interfaces" and the list "interface" elements to be versioned elements, the server's response to the request above might look like:

```
<rpc-reply message-id="1"
           xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
           xmlns:txid="urn:ietf:params:xml:ns:netconf:txid:1.0">
  <data txid:etag="def88884321">
    <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
                txid:etag="def88884321">
      <interface txid:etag="def88884321">
        <name>GigabitEthernet-0/0</name>
        <description>Management Interface</description>
        <type>ianaift:ethernetCsmacd</type>
        <enabled>true</enabled>
      </interface>
      <interface txid:etag="abc12345678">
        <name>GigabitEthernet-0/1</name>
        <description>Upward Interface</description>
        <type>ianaift:ethernetCsmacd</type>
        <enabled>true</enabled>
      </interface>
    </interfaces>
    <nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm"/>
      <groups>
        <group>
          <name>admin</name>
          <user-name>sakura</user-name>
          <user-name>joe</user-name>
        </group>
      </groups>
    </nacm>
  </data>
</rpc>
```

4.2.2. Configuration Response Pruning

A NETCONF client that already knows some etag values MAY request that the configuration retrieval request is pruned with respect to the client's prior knowledge.

To retrieve only changes for "ietf-interfaces" that do not have the last known etag value "abc12345678", but include the entire configuration for "nacm", regardless of etags, a client might send:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1"
     xmlns:txid="urn:ietf:params:xml:ns:netconf:txid:1.0">
  <get-config>
    <source>
      <running/>
    </source>
    <filter>
      <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
        txid:etag="abc12345678"/>
      <nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm"/>
    </filter>
  </get-config>
</rpc>
```

When a NETCONF server receives a get-config or get-data request containing an element with a client specified etag attribute, there are several different cases:

- The element is not a versioned element, i.e. the server does not maintain an etag value for this element. In this case, the server MUST look up the closest ancestor that is a versioned element, and proceed as if the client had specified the etag value for that element.

- The element is a versioned element, and the client specified etag attribute value is different than the server's etag value for this element. In this case the server MUST return the contents as it would otherwise have done, adding the etag attributes of all child versioned elements to the response. In case the client has specified etag attributes for some child elements, then these cases MUST be re-evaluated for those elements.

- The element is a versioned element, and the client specified etag attribute value matches the server's etag value. In this case the server MUST return the element decorated with an etag attribute with the value "=", and child elements pruned.

For list elements, pruning child elements means that key elements MUST be included in the response, and other child elements MUST NOT be included. For containers, child elements MUST NOT be included.

For example, assuming the NETCONF server configuration is the same as in the previous rpc-reply example, the server's response to request above might look like:

```
<rpc-reply message-id="1"
           xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
           xmlns:txid="urn:ietf:params:xml:ns:netconf:txid:1.0">
  <data txid:etag="def88884321">
    <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
                txid:etag="def88884321">
      <interface txid:etag="def88884321">
        <name>GigabitEthernet-0/0</name>
        <description>Management Interface</description>
        <type>ianaift:ethernetCsmacd</type>
        <enabled>true</enabled>
      </interface>
      <interface txid:etag="=">
        <name>GigabitEthernet-0/1</name>
      </interface>
    </interfaces>
    <nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm"/>
      <groups>
        <group>
          <name>admin</name>
          <user-name>sakura</user-name>
          <user-name>joe</user-name>
        </group>
      </groups>
    </nacm>
  </data>
</rpc>
```

4.3. Configuration Update

Whenever the configuration on a server changes for any reason, the server MUST update the etag value for all versioned elements that have children that changed.

If the change is due to a NETCONF client edit-config or edit-data request that includes the ietf-netconf-txid:with-etag presence container, the server MUST return the etag value of the targeted datastore as an attribute on the XML ok tag in the rpc-reply.

The server MUST NOT change the etag value of a versioned element unless a child element of that element has been changed. The server MUST NOT change any etag values due to changes in config false data.

How the server selects a new etag value to use for the changed elements is described in section ETag attribute (Section 4.1).

For example, if a client wishes to update the interface description for interface "GigabitEthernet-0/1" to "Downward Interface", it might send:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <edit-config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
               xmlns:ietf-netconf-txid=
                "urn:ietf:params:xml:ns:yang:ietf-netconf-txid">
    <target>
      <candidate/>
    </target>
    <test-option>test-then-set</test-option>
    <ietf-netconf-txid:with-etag/>
    <config>
      <interfaces
        xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
        <interface>
          <name>GigabitEthernet-0/1</name>
          <description>Downward Interface</description>
        </interface>
      </interfaces>
    </config>
  </edit-config>
</rpc>
```

The server would update the description leaf in the candidate datastore, and return an rpc-reply as follows:

```
<rpc-reply message-id="1"
           xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
           xmlns:txid="urn:ietf:params:xml:ns:netconf:txid:1.0">
  <ok txid:etag="ghi55550101"/>
</rpc-reply>
```

A subsequent get-config request for "ietf-interfaces", with txid:etag="?" might then return:

```
<rpc-reply message-id="1"
           xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
           xmlns:txid="urn:ietf:params:xml:ns:netconf:txid:1.0">
  <data txid:etag="ghi55550101">
    <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
                txid:etag="ghi55550101">
      <interface txid:etag="def88884321">
        <name>GigabitEthernet-0/0</name>
        <description>Management Interface</description>
        <type>ianaift:ethernetCsmacd</type>
        <enabled>true</enabled>
      </interface>
      <interface txid:etag="ghi55550101">
        <name>GigabitEthernet-0/1</name>
        <description>Downward Interface</description>
        <type>ianaift:ethernetCsmacd</type>
        <enabled>true</enabled>
      </interface>
    </interfaces>
  </data>
</rpc>
```

In case the server at this point received a configuration change from another source, such as a CLI operator, adding an MTU value for the interface "GigabitEthernet-0/0", a subsequent get-config request for "ietf-interfaces", with txid:etag="?" might then return:

```
<rpc-reply message-id="1"
           xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
           xmlns:txid="urn:ietf:params:xml:ns:netconf:txid:1.0">
  <data txid:etag="cli22223333">
    <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
                txid:etag="cli22223333">
      <interface txid:etag="cli22223333">
        <name>GigabitEthernet-0/0</name>
        <description>Management Interface</description>
        <type>ianaift:ethernetCsmacd</type>
        <enabled>true</enabled>
        <mtu>768</mtu>
      </interface>
      <interface txid:etag="ghi55550101">
        <name>GigabitEthernet-0/1</name>
        <description>Downward Interface</description>
        <type>ianaift:ethernetCsmacd</type>
        <enabled>true</enabled>
      </interface>
    </interfaces>
  </data>
</rpc>
```

4.3.1. Conditional Configuration Update

When a NETCONF client sends an edit-config or edit-data request to a NETCONF server that implements this specification, the client MAY specify expected etag values on the versioned elements touched by the transaction.

If such an etag value differs from the etag value stored on the server, the server MUST reject the transaction and return an rpc-error as specified in section Conditional Transactions (Section 3.2).

Additionally, the error-info tag MUST contain an sx:structure etag-value-mismatch-error-info as defined in the module ietf-netconf-txid, with mismatch-path set to the instance identifier value identifying one of the versioned elements that had an etag value mismatch, and mismatch-etag-value set to the server's current value of the etag attribute for that versioned element.

For example, if a client wishes to delete the interface "GigabitEthernet-0/1" if and only if its configuration has not been altered since this client last synchronized its configuration with the server (at which point it received the etag "ghi55550101"), regardless of any possible changes to other interfaces, it might send:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1"
     xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
     xmlns:txid="urn:ietf:params:xml:ns:netconf:txid:1.0"
     xmlns:ietf-netconf-txid=
       "urn:ietf:params:xml:ns:yang:ietf-netconf-txid">
  <edit-config>
    <target>
      <candidate/>
    </target>
    <test-option>test-then-set</test-option>
    <ietf-netconf-txid:with-etag/>
    <config>
      <interfaces
        xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
        <interface nc:operation="delete"
                   txid:etag="ghi55550101">
          <name>GigabitEthernet-0/1</name>
        </interface>
      </interfaces>
    </config>
  </edit-config>
</rpc>
```

If interface "GigabitEthernet-0/1" has the etag value "ghi55550101", as expected by the client, the transaction goes through, and the server responds something like:

```
<rpc-reply message-id="1"
           xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
           xmlns:txid="urn:ietf:params:xml:ns:netconf:txid:1.0">
  <ok txid:etag="xyz77775511"/>
</rpc-reply>
```

A subsequent get-config request for "ietf-interfaces", with txid:etag="?" might then return:

```
<rpc-reply message-id="1"
           xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
           xmlns:txid="urn:ietf:params:xml:ns:netconf:txid:1.0">
  <data txid:etag="xyz77775511">
    <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
                txid:etag="xyz77775511">
      <interface txid:etag="def88884321">
        <name>GigabitEthernet-0/0</name>
        <description>Management Interface</description>
        <type>ianaift:ethernetCsmacd</type>
        <enabled>true</enabled>
      </interface>
    </interfaces>
  </data>
</rpc-reply>
```

In case interface "GigabitEthernet-0/1" did not have the expected etag value "ghi55550101", the server rejects the transaction, and might send:

```
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
           xmlns:if="urn:ietf:params:xml:ns:yang:ietf-interfaces"
           xmlns:ietf-netconf-txid=
             "urn:ietf:params:xml:ns:yang:ietf-netconf-txid">
           message-id="1">
  <rpc-error>
    <error-type>protocol</error-type>
    <error-tag>operation-failed</error-tag>
    <error-severity>error</error-severity>
    <error-info>
      <ietf-netconf-txid:etag-value-mismatch-error-info>
        <ietf-netconf-txid:mismatch-path>
          /if:interfaces/if:interface[if:name="GigabitEthernet-0/0"]
        </ietf-netconf-txid:mismatch-path>
        <ietf-netconf-txid:mismatch-etag-value>
          cli22223333
        </ietf-netconf-txid:mismatch-etag-value>
      </ietf-netconf-txid:etag-value-mismatch-error-info>
    </error-info>
  </rpc-error>
</rpc-reply>
```

4.4. ETags with Other NETCONF Operations

The following NETCONF Operations also need some special considerations.

discard-changes The server MUST ensure the etag attributes in the candidate datastore get the same values as in the running datastore when this operation runs.

copy-config The server MUST ensure the etag attributes retain the same values as in the soruce datastore.

If copy-config is used to copy from a source that is not a datastore, the server MUST ensure etags are given new values.

delete-config The server MUST ensure the datastore etag is given a new value.

commit At commit, with regards to the etag values, the server MUST treat the contents of the candidate datastore as if any etag attributes provided by the client were provided in a single edit-config towards the running datastore. If the commit is rejected due to etag mismatch, the rpc-error message specified in section Conditional Configuration Update (Section 4.3.1) MUST be sent.

The client MAY request that the new etag value is returned as an attribute on the ok response for a successful commit. The client requests this by adding with-etag to the commit operation.

For example, a client might send:

```
<rpc message-id="1"
    xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    xmlns:ietf-netconf-txid=
      "urn:ietf:params:xml:ns:yang:ietf-netconf-txid"
  <commit>
    <ietf-netconf-txid:with-etag/>
  </commit>
</rpc>
```

Assuming the server accepted the transaction, it might respond:

```
<rpc-reply message-id="1"
    xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
    xmlns:txid="urn:ietf:params:xml:ns:netconf:txid:1.0">
  <ok txid:etag="ghi55550101"/>
</rpc-reply>
```

5. YANG Modules

```
module ietf-netconf-txid {
  yang-version 1.1;
  namespace
    'urn:ietf:params:xml:ns:yang:ietf-netconf-txid';
  prefix ietf-netconf-txid;

  import ietf-netconf {
    prefix nc;
  }

  import ietf-netconf-nmda {
    prefix ncds;
  }

  import ietf-yang-structure-ext {
    prefix sx;
  }

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

  contact
    "WG Web:   <http://tools.ietf.org/wg/netconf/>
     WG List:  <netconf@ietf.org>

     Author:   Jan Lindblad
               <mailto:jlindbla@cisco.com>";

  description
    "NETCONF Transaction ID aware operations for NMDA.

     Copyright (c) 2021 IETF Trust and the persons identified as
     the document authors.  All rights reserved.

     Redistribution and use in source and binary forms, with or
     without modification, is permitted pursuant to, and subject
     to the license terms contained in, the Simplified BSD License
     set forth in Section 4.c of the IETF Trust's Legal Provisions
     Relating to IETF Documents
     (http://trustee.ietf.org/license-info).

     This version of this YANG module is part of RFC XXXX; see
     the RFC itself for full legal notices.";

  revision 2021-11-01 {
    description
      "Initial revision";
    reference
      "RFC XXXX: Xxxxxxxxx";
  }
```

```
typedef etag-t {
  type string {
    pattern ".* .*" {
      modifier invert-match;
    }
    pattern ".*\".*" {
      modifier invert-match;
    }
    pattern ".*\\.*" {
      modifier invert-match;
    }
  }
  description
    "Unique Entity-tag value representing a specific transaction.
     Could be any string that does not contain spaces, double
     quotes or backslash.  The values '?' and '=' have special
     meaning.";
}

grouping transaction-id-grouping {
  container with-etag {
    presence
      "Indicates that the client requests the server to include a
       txid:etag transaction id in the rpc-reply";
  }
  description
    "Grouping for transaction id mechanisms, to be augmented into
     rpcs that modify configuration data stores.";
}

augment /nc:edit-config/nc:input {
  uses transaction-id-grouping;
  description
    "Injects the transaction id mechanisms into the
    edit-config operation";
}

augment /nc:commit/nc:input {
  uses transaction-id-grouping;
  description
    "Injects the transaction id mechanisms into the
    commit operation";
}

augment /ncds:edit-data/ncds:input {
  uses transaction-id-grouping;
  description
    "Injects the transaction id mechanisms into the
```

```
      edit-data operation";

  sx:structure etag-value-mismatch-error-info {
    container etag-value-mismatch-error-info {
      description
         "This error is returned by a NETCONF server when a client
          sends a configuration change request, with the additonal
          condition that the server aborts the transaction if the
          server's configuration has changed from what the client
          expects, and the configuration is found not to actually
          not match the client's expectation.";
      leaf mismatch-path {
        type instance-identifier;
        description
          "Indicates the YANG path to the element with a mismatching
           etag value.";
      }
      leaf mismatch-etag-value {
        type etag-t;
        description
          "Indicates server's value of the etag attribute for one
           mismatching element.";
      }
    }
  }
}
```

6. Security Considerations

TODO Security

7. IANA Considerations

This document registers the following capability identifier URN in the 'Network Configuration Protocol (NETCONF) Capability URNs' registry:

urn:ietf:params:netconf:capability:txid:1.0

This document registers two XML namespace URNs in the 'IETF XML registry', following the format defined in RFC 3688.

URI: urn:ietf:params:xml:ns:netconf:txid:1.0

URI: urn:ietf:params:xml:ns:yang:ietf-netconf-txid

Registrant Contact: The NETCONF WG of the IETF.

XML: N/A, the requested URIs are XML namespaces.

This document registers one module name in the 'YANG Module Names' registry, defined in RFC 6020.

name: ietf-netconf-txid

prefix: ietf-netconf-txid

namespace: urn:ietf:params:xml:ns:yang:ietf-netconf-txid

RFC: XXXX

8. Changes

8.1. Major changes in -01 since -00

*Updated the text on numerous points in order to answer questions that appeared on the mailing list.

*Changed the document structure into a general transaction id part and one etag specific part.

*Renamed entag attribute to etag, prefix to txid, namespace to urn:ietf:params:xml:ns:yang:ietf-netconf-txid.

*Set capability string to urn:ietf:params:netconf:capability:txid:1.0

*Changed YANG module name, namespace and prefix to match names above.

*Harmonized/slightly adjusted etag value space with RFC 7232 and RFC 8040.

*Removed all text discussing etag values provided by the client (although this is still an interesting idea, if you ask the author)

*Clarified the etag attribute mechanism, especially when it comes to matching against non-versioned elements, its cascading upwards in the tree and secondary effects from when- and choice-statements.

*Added a mechanism for returning the server assigned etag value in get-config and get-data.

*Added section describing how the NETCONF discard-changes, copy-config, delete-config and commit operations work with respect to etags.

*Added IANA Considerations section.

*Removed all comments about open questions.

9. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

Acknowledgments

The author wishes to thank Benoit Claise for making this work happen, and the following individuals, who all provided helpful comments: Per Andersson, Kent Watsen, Andy Bierman, Robert Wilton, Qiufang Ma.

Author's Address

Jan Lindblad
Cisco Systems

Email: jlindbla@cisco.com