Workgroup: NETCONF
Internet-Draft:
draft-lindblad-netconf-transaction-id-00
Published: 2 November 2020
Intended Status: Standards Track
Expires: 6 May 2021
Authors: J. Lindblad
Cisco Systems

Transaction ID Mechanism for NETCONF

Abstract

TODO Abstract

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 6 May 2021.

Copyright Notice

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

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, which arrives soon thereafter.

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 RFC8040, 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 RFC7232 "Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests".

This document defines similar functionality for NETCONF, RFC 6241 RFC6241.

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 Extension

This document describes a NETCONF extension which modifies the behavior of get-config, get-data, edit-config and edit-data such that clients are able to conditionally retrieve and update the configuration in a NETCONF server. NETCONF servers that supports this extension MUST announce the capability "FIXME".

The extended operations defined in this document pertains to YANG containers and list elements. It is NOT REQUIRED that a conforming server allows the extended operations to apply to all containers and list elements in the server configuration. The set of containers and list elements that the server supports with respect to this NETCONF extension are collectively referred to as the "versioned elements".

The NETCONF server will maintain a record of the transaction that last changed each versioned element. This transaction-id meta level data is communicated between the server and client in the form of an XML attribute called "entag". The values for the entag attribute is up to the clients and servers to decice as opaque quantities. It is essential that the entag values have a large value space in order to not run out or collide. They SHOULD be at least 32-bit quantities.

Entag attribute values are encoded as YANG strings.

> Comment, to be removed
>
> Do we want to limit the entag attribute strings in some way? E.g. only base64 characters, some min or max length, ...?

4. Configuration Retreival

When a NETCONF client retrieves the configuration from a NETCONF server that implement this specification, it may request that the configuration is entity tagged. The entity tags are XML attributes added to some of the retrieved configuration elements by the server. These elements are collectively called the "versioned elements".

The entity-tag (entag) attributes are guaranteed to change every time there has been a configuration change at or below the element bearing the attribute.

Clients request entity tags to be added by setting the ietf-netconf-transaction-id:entag attribute to the value "?" on one or more elements in the request. Entags MUST be returned for all descendant versioned elements. In order to request that entags are returned for the entire configuration, the client can place the attribute on the top edit-config or edit-data tags. For more specific retrieval, the client inserts entag attributes in the filter section.

To retrieve entag 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:ietf-netconf-transaction-id=
       "FIXME">
  <get-config ietf-netconf-transaction-id:entag="?"/>
</rpc>
```

To retrieve entag 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:ietf-netconf-transaction-id=
       "FIXME">
  <get-config>
    <source>
      <running/>
    </source>
    <filter>
      <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
        ietf-netconf-transaction-id:entag="?"/>
      <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 ietf-netconf-transaction-id:entag attributes with the value "?", it MUST return entag attributes for all versioned elements below this point included in the reply.

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:ietf-netconf-transaction-id=
             "FIXME">
  <data ietf-netconf-transaction-id:entag="def88884321">
    <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
                ietf-netconf-transaction-id:entag="def88884321">
      <interface ietf-netconf-transaction-id:entag="def88884321">
        <name>GigabitEthernet-0/0</name>
        <description>Management Interface</description>
        <type>ianaift:ethernetCsmacd</type>
        <enabled>true</enabled>
      </interface>
      <interface ietf-netconf-transaction-id:entag="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>
```

4.1. Configuration Response Pruning

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

By specifying the previously received entag attribute values in the get-config or get-data request, the client indicates that child elements of already known parts of the configuration SHALL be omitted.

To retrieve only changes for "ietf-interfaces" since the last known transaction-id "abc12345678", but include the entire configuration for "nacm", a client might send:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1"
     xmlns:ietf-netconf-transaction-id=
       "FIXME">
  <get-config>
    <source>
      <running/>
    </source>
    <filter>
      <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
        ietf-netconf-transaction-id:entag="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 ietf-netconf-transaction-id:entag attributes with the same value as the entag value known by the server for that element, it MUST prune the contents of that subtree.

In case the element with a matching entag value is a container, the container tag is returned with an entag attribute value of "=". No child elements are returned for the container.

In case the element with a matching entag value is a list element, the list element tag is returned with an entag attribute value of "=". The list element will include the list elemenet keys, but no other child elements.

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:ietf-netconf-transaction-id=
             "FIXME">
  <data ietf-netconf-transaction-id:entag="def88884321">
    <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
                ietf-netconf-transaction-id:entag="def88884321">
      <interface ietf-netconf-transaction-id:entag="def88884321">
        <name>GigabitEthernet-0/0</name>
        <description>Management Interface</description>
        <type>ianaift:ethernetCsmacd</type>
        <enabled>true</enabled>
      </interface>
      <interface ietf-netconf-transaction-id:entag="=">
        <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>
```

5. 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 a transaction-id value.

If specified, the server MUST use this value as the new value for all entag attribute values of any versioned element touched by the transaction, if and only if the operation is successful. The entag value must be updated regardless of whether an actual value change took place or not. An element is touched if it is mentioned in the transaction, even if it merely sets the element to the same value it already has.

If the server side configuration changes for any reason, and there is no transaction-id value specified by a client, servers that supports this specification MUST update the entag values as if a NETCONF

client had made the change and specified a transaction-id. In this case, the server MUST choose a random transaction-id value to use.

Comment, to be removed

Is talk about "random" good enough, or do we need to get specific?

Every time a versioned element has its entag value updated, the same value must be set to all parent versioned elements' entag attributes, cascading all the way to the datastore root.

For example, if a client wishes to update the interface description for interface "GigabitEthernet-0/1" to "Downward Interface", under transaction-id "ghi55550101", it might send:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1"
     xmlns:ietf-netconf-transaction-id=
       "FIXME">
  <edit-config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <target>
      <candidate/>
    </target>
    <test-option>test-then-set</test-option>
    <ietf-netconf-transaction-id:transaction-id>
      ghi55550101
    </transaction-id>
    <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>
```

A subsequent get-config request for "ietf-interfaces", with ietf-netconf-transaction-id:entag="?" might then return:

```
<rpc-reply message-id="1"
           xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
           xmlns:ietf-netconf-transaction-id=
             "FIXME">
  <data>
    <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
                ietf-netconf-transaction-id:entag="ghi55550101">
      <interface ietf-netconf-transaction-id:entag="def88884321">
        <name>GigabitEthernet-0/0</name>
        <description>Management Interface</description>
        <type>ianaift:ethernetCsmacd</type>
        <enabled>true</enabled>
      </interface>
      <interface ietf-netconf-transaction-id:entag="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 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 ietf-netconf-transaction-id:entag="?" might then return:

```
<rpc-reply message-id="1"
           xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
           xmlns:ietf-netconf-transaction-id=
             "FIXME">
  <data ietf-netconf-transaction-id:entag="auto22223333">
    <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
                ietf-netconf-transaction-id:entag="auto22223333">
      <interface ietf-netconf-transaction-id:entag="auto22223333">
        <name>GigabitEthernet-0/0</name>
        <description>Management Interface</description>
        <type>ianaift:ethernetCsmacd</type>
        <enabled>true</enabled>
        <mtu>768</mtu>
      </interface>
      <interface ietf-netconf-transaction-id:entag="ghi55550101">
        <name>GigabitEthernet-0/1</name>
        <description>Downward Interface</description>
        <type>ianaift:ethernetCsmacd</type>
        <enabled>true</enabled>
      </interface>
    </interfaces>
  </data>
</rpc>
```

5.1. Conditional Configuration Update

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

Even if a client is constantly connected to a device, and even possibly receiving notifications when a server device's configuration changes, there is always a possibility that a change is introduced by another party in the time window between when the client last received an update about the server's configuration until the server executes a configuration change request.

By leveraging conditional transactions, this race condition can be eliminated efficiently. If the client provides the transaction-id it expects the device to have as part of it's configuration change request, and the device guarantees to only execute the request in case the transaction-id in the request matches that on the server, the race condition is removed.

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 entag values on the versioned elements touched by the transaction.

If such an entag value differs from the entag value stored on the server, the server MUST reject the transaction.

If the server rejects the transaction because the configuration entag value differs from the client's expectation, ther 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 MUST contain a sx:structure entag-value-mismatch-error-info, with mismatch-path set to the instance identifier value identifying one of the versioned elements that had an entag value mismatch, and mismatch-entag-value set to the server's current value of the entag 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 a transaction-id "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:ietf-netconf-transaction-id=
       "FIXME">
  <edit-config>
    <target>
      <candidate/>
    </target>
    <test-option>test-then-set</test-option>
    <config>
      <interfaces
        xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
        <interface nc:operation="delete"
                   ietf-netconf-transaction-id:entag="ghi55550101">
          <name>GigabitEthernet-0/1</name>
        </interface>
      </interfaces>
    </config>
  </edit-config>
</rpc>
```

If interface "GigabitEthernet-0/1" has the entag value "ghi55550101", as expected by the client, the transaction goes through.

A subsequent get-config request for "ietf-interfaces", with ietf-netconf-transaction-id:entag="?" might then return:

```
<rpc-reply message-id="1"
           xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
           xmlns:ietf-netconf-transaction-id=
             "FIXME">
  <data ietf-netconf-transaction-id:entag="auto77775511">
    <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
                ietf-netconf-transaction-id:entag="auto77775511">
      <interface ietf-netconf-transaction-id:entag="def88884321">
        <name>GigabitEthernet-0/0</name>
        <description>Management Interface</description>
        <type>ianaift:ethernetCsmacd</type>
        <enabled>true</enabled>
      </interface>
    </interfaces>
  </data>
</rpc>
```

If interface "GigabitEthernet-0/1" did not have the entag 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-transaction-id=
             "FIXME">
           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-transaction-id:entag-value-mismatch-error-info>
        <ietf-netconf-transaction-id:mismatch-path>
          /if:interfaces/if:interface[if:name="GigabitEthernet-0/0"]
        </ietf-netconf-transaction-id:mismatch-path>
        <ietf-netconf-transaction-id:mismatch-entag-value>
          auto77775511
        </ietf-netconf-transaction-id:mismatch-entag-value>
      </ietf-netconf-transaction-id:entag-value-mismatch-error-info>
    </error-info>
  </rpc-error>
</rpc-reply>
```

Comment, to be removed

In order to reach the full flexibility with the above transaction rejection mechanism, it might make sense to reference parts of the configuration just to see that they have not moved, with no intent to make changes there. To support this use case, a new operation mode "nocreate" might be useful. This would allow an edit config to talk about parts of the configuration which are expected to exist with a particular confiuguration, and to abort the transaction if they do not exist.

Comment, to be removed

NETCONF clients may be equally interested to apply a mechanism similar to entags when retrieving operational state as well, since there is often vey much of this data, and some if changes rather rarely. To support this use case, some sort of server maintained change indicators may be invented, and combined with a similar retrieval filter.

6. YANG Modules

Comment, to be removed

This is YANG 1.1. Do we also want 1.0? Makes it possible to implement on 1.0 servers

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

  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) 2020 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 2020-10-01 {
    description
      "Initial revision";
    reference
      "RFC XXXX: Xxxxxxxxx";
  }
```

```
typedef transaction-id-t {
  type string;
  description
    "Unique value representing a specific transaction";
}

grouping transaction-id-grouping {
  leaf transaction-id {
    type transaction-id-t;
    description
      "Transaction-id value selected by the client.  This string
       should be chosen to give a high probability to be unique on
       the server.";
  }
  description
    "Grouping for transaction-id, 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 leaf into the edit-config
    operation";
}

augment /ncds:edit-data/ncds:input {
  uses transaction-id-grouping;
  description
    "Injects the transaction-id leaf into the edit-data
    operation";
}

sx:structure entag-value-mismatch-error-info {
  container entag-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
         entag value.";
    }
```

```
      leaf mismatch-entag-value {
        type transaction-id-t;
        description
          "Indicates server's value of the entag attribute for one
           mismatching element.";
      }
    }
  }
}
```

7. Security Considerations

TODO Security

8. IANA Considerations

This document has no IANA actions.

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/info/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/info/rfc8174>.

Acknowledgments

TODO acknowledge.

Author's Address

Jan Lindblad
Cisco Systems

Email: jlindbla@cisco.com