A Transaction Test Module for the NETCONF Verify Operation
draft-cole-netconf-transaction-00

Abstract

This document extends the capabilities of the NETCONF configuration management protocol in order to standardize mechanisms to perform sets of active tests (i.e., verification) against servers' running configuration to afford the client and server a more robust and resilient configuration management capability. Specifically, this document defines a transaction test module based upon the defined set of Uniform Resource Locators. The transaction tests in this module are executed by the NETCONF verify operation.

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 http://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 January 6, 2011.

Copyright Notice

Copyright (c) 2010 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of
publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must 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 ......................................................... 3
   1.1. Benefits of This Work ......................................... 5
   1.2. Requirements Language ....................................... 5
   1.3. Outline ......................................................... 5
2. The Transaction Test Module ......................................... 6
   2.1. Verify Capability ............................................ 6
   2.2. Transaction Test Module Construction and Use ............... 6
3. IANA Considerations ................................................. 7
4. Security Considerations ............................................. 7
5. References .......................................................... 8
   5.1. Normative References ......................................... 8
   5.2. Informative References ....................................... 9
Appendix A. transaction.yang Module ................................. 9
Authors' Addresses ................................................... 21
1. Introduction

This document identifies enhancements to NETCONF capabilities to achieve a more robust model of configuration management for future IETF systems. Most network management systems which are required to provide a highly robust network service rely upon some form of out-of-band access for configuration management. This provides an alternative management entry into devices in the event that in-band access is unavailable due to, e.g., mis-configuration. However, not all network deployments can afford the luxury of alternative networks for management access to all networking devices, nor should this be necessary. Examples include Mobile Ad-Hoc Wireless Networks (MANETs) and other forms of Disruption Tolerant Networks (DTNs). All managed networks, as well, would benefit from a more robust and extensive configuration management capability from the IETF, e.g., to provide equivalent network reliability at reduced infrastructure costs.

Towards this objective, we propose that the NETCONF protocol RFC 4741 [RFC4741] requires extension of capabilities to define and manage active tests and assess success, i.e., Verification, (from both the client and the servers) involving server-side running configuration. This document augments the verify capability within NETCONF by defining a transaction test module. This allows the network management application to exercise the transaction tests through a standard mechanism. In this test module, the transactions are defined within the context of defined Uniform Resource Locators (URLs). This allows the network management application to exercise the transaction tests through an extensible mechanism.

As an example, we envision a NETCONF client-server interaction model shown in the below figure. Here, the client issues a \texttt{<commit>} with the confirming option. As part of testing prior to issuing the confirming \texttt{<commit>} the client wishes to execute a set of verification transaction tests from the server. It issues the \texttt{<verify>} operation to manage this aspect of verification transaction testing. The client passes a reference to the server indicating instances of specific pre-configured transaction tests within this
module that define the specific test suite. The server executes these as part of the NETCONF <verify> testing process.
Simultaneously, the client may also run a set of tests to gain confidence in the proposed configuration changes to the server. Once the server completes its test execution, it indicates success through notification messages. Once the client is comfortable with its own tests and those of the server, it issues the confirming <commit> to the server which forces the server to commit to the proposed configuration change; else the server backs out of the proposed configuration changes.
NETCONF defines the term 'validation' as the set of checks performed on proposed configuration code up to the point that the server places it into its running-configuration. We use the term 'verification' as the act of performing active tests against configuration code in the running-configuration on the server. Verification tests can be executed from either the NETCONF client or the NETCONF server, or from a NETCONF server(a) against running configuration code on a NETCONF server(b), or all combinations.

In this document, we define the transaction.yang module as a first example of a test module supporting the NETCONF verify operation. This allows for extensible verification testing of configuration across the base of IETF compliant devices. This leads to more resilient configuration management for operators managing multi-vendor networks of devices. This will promote future integrated network management capabilities as opposed to device management capabilities.

1.1. Benefits of This Work

Our objective is to promote the development of a robust and resilient network configuration capability, building upon the improvements afforded by the NETCONF protocol and its associated modeling language, YANG [YANG].

The envisioned benefits of a standardized set of mechanisms and
capabilities for verification testing include:

- Minimize faulty configuration and network disconnects,
- Provide for uniform methods for control, execution and reporting of verification testing in multi-vendor networks,
- Improve automation of extensive verification testing,
- Provide opportunity for device modelers to associate/recommend tests tied to specific configuration items, and
- Improve efficiency of coordinated network upgrades.

### 1.2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC 2119](https://tools.ietf.org/html/rfc2119).

### 1.3. Outline

In the remainder of this document we present a description of the transaction.yang test module. This is followed with 'Acknowledgments' and 'IANA Considerations' sections. A section on 'Security Considerations' is provided concluding the main body of the document. In the appendix, i.e., 'Appendix A: transaction.yang', we define the transaction.yang module.

### 2. The Transaction Test Module

The transaction.yang module defines a set of transaction tests that can be instrumented via NETCONF and executed through the verify operation. We briefly discuss the verify operation in the context of executing the transaction tests. We then discuss the construction of the transaction.yang module. The definitive definition of the transaction.yang module is found in Appendix A of this document.

#### 2.1. Verify Capability
The verify operation, defined in VERIFY [VERIFY], allows for the execution of verification tests within the NETCONF protocol. The construction of the verify operation is illustrated in the following diagram. Here a verify command is given with associated timeout and test-template parameters. The multiple test-template parameters each indicate a specific set of tests defined within the transaction.yang module resident on the server. The specific tests are pre-configured through standard NETCONF commands prior to issuing the verify operation. The definition of the verify operation allows various levels of reporting of the test results back to the NETCONF client.

```xml
<rpc xmlns="netconf-base" message-id="101">
  <verify xmlns="verify-module">
    <timeout>3600</timeout>
    <test-template xmlns:as="transaction-module">
      /tt:transaction/tt:controlTableEntry[tt:controlTableIndex=21]
      /tt:transaction/tt:controlTableEntry[tt:controlTableIndex=42]
      /tt:transaction/tt:controlTableEntry[tt:controlTableIndex=48]
    </test-template>
    <verifyStatus>true</verifyStatus>
    <extendedStatus>false</extendedStatus>
  </verify>
</rpc>
```

Figure 2

2.2. Transaction Test Module Construction and Use

The transaction.yang module is designed to support an extensible set of transaction test for the purpose of verification testing of proposed configuration changes. As such, we have modeled the module after the Uniform Resource Locator (URL) definition. The module is defined in six basic functions:

- **Protocol** - defines the set of protocol transactions supported by the server and referenced through the URL 'scheme'.
- **Location Profile** - defines a set of URLs which are predefined for later execution.
- Network Profile - defines a set of reuse-able network layer parameters.
- Metric Profile - defines the performance aspects of the tests, e.g., frequency, metric, etc.
- Control Table - defines the specific verification test sets.
- Results table - contains the results of the verification test sets.

Refer to Appendix A for the definitive statement of the transaction.yang module.

3. IANA Considerations

This memo includes no request to IANA.

All drafts are required to have an IANA considerations section (see the update of RFC 2434 [I-D.narten-iana-considerations-rfc2434bis] for a guide). If the draft does not require IANA to do anything, the section contains an explicit statement that this is the case (as above). If there are no requirements for IANA, the section will be removed during conversion into an RFC by the RFC Editor.

4. Security Considerations

This section presents the required security considerations for all IETF protocols and capabilities. This section was developed following guidelines within RFC 3552 [RFC3552].

This section addresses the security concerns and objectives for the for the use of the transaction.yang module within the context of the :verify capability in NETCONF. (NOTE: This section is currently TBD.)

Security issues related to the use of the transaction.yang module should address issues specific to the remote execution of verification tests. Here is an initial list of potential
considerations:

- Verification requires server-side tests that require that packets to be injected into the network for the purpose of measuring some performance characteristics. As such, associated test modules will contain sensitive network and application data; e.g., user IDs and passwords. Further, if security is compromised, this capability could provide a source for denial-of-service, and potential other, attacks.

- The configuration of verification tests may require passing sensitive network information. For this reason, this configuration information should be encrypted prior to transport over the network.

- Some test attributes configure username and password information for some application-level protocols as indicated above. Access to these attributes may provide unauthorized use of resources.

- Some test attributes configure the size and rate of traffic flows for the purpose of performance measurements. Access to these attributes may exacerbate the use of this capability in denial-of-service attacks. It is recommended that test modules define a maximum packet rate on the device and to indicate this rate. Other objects that control aspects of the test packets related to packet size and rate are will exist in test modules and bounds on these should be set.

- Test module objects will exist which set the source and destination addresses on the packet headers. The server should not allow the setting of source addresses on the test packets other than those that are administratively configured onto the server.

5. References

5.1. Normative References


5.2. Informative References


Appendix A. transaction.yang Module

In this appendix we define the transaction.yang model for use in conjunction with the robust-netconf capabilities.

========Contents of "transaction.yang"=========

module transaction {

    namespace "unassigned";
    prefix "tt";

    import ietf-yang-types { prefix yang; }
    import ietf-inet-types { prefix inet; }

    organization "IETF";

    contact
        "Andy Bierman
        InterWorking Labs
        EMail: andyb@iwl.com

        Robert G. Cole
description
"The module for entities implementing the transaction test set in support of the NETCONF verify capability."

revision 2010-05-07 {

description "Zeroth revision: Initial version of the transaction testing module. This is modeled after the draft ping.yang module from draft-cole-netconf-verify-00.txt and from the definition of Uniform Resource Locators (URLs) [RFC 1738].

This module allows a management agent to instrument and execute a broad set of protocol transactions in order to perform a broad range of connectivity tests. These tests, executed in conjunction with the NETCONF verify operation, can be used to provide a robust configuration change capability. This capability is described in draft-cole-netconf-verify-00.txt.";
}
list transactionProtocolEntry {
    key "transactionProtocolIndex";
    config false;

    leaf transactionProtocolIndex {
        type uint32;
        description
    }

    leaf transactionProtocolScheme {
        type string;
        description
            "Identifies a specific protocol
            transaction supported by this device.
            The transaction protocol is defined
            in the definition of the 'scheme'
            of the associated URL. These are
            registered by IANA [RFC 4395].";
    }

    leaf protocolReference {
        type string;
        config false;
        description
            "URL for the definition of this
            URL scheme. This could be a reference
            to an RFC or to a publically
            available reference.";
    }
}

-- ends the transactionProtocolEntry list --
The Location Profile defines a set of URLs which are pre-defined in the server for the purpose of executing verification tests controlled by the NETCONF verify operation.

```
list locationProfileEntry {
    key "locationProfileIndex";
    config true;

    leaf locationProfileIndex {
        type uint32;
        description "Identifies the specific URL to be accessed by execution of the transaction test.";
    }

    leaf locationProfileSchemeIndex {
        type uint32;
        description "Contains the integer referencing the transactionProtocolIndex in the capabilities set found in the transactionProtocolEntry in this module.";
    }

    leaf locationProfileUser {
        type string;
        description "The username associated with the URL defined within this locationProfileEntry. Some URLs do not allow user entries, in which case this string should be NULL.";
    }

    leaf locationProfilePassword {
        type string;
    }
}
```
description
"The password associated with the URL defined within this locationProfileEntry.
If the specific scheme associated with this URL does not allow user and password,
then this string should be set to NULL."
}

leaf locationProfileHost {
  type string;
  description
    "The fully qualified domain name of a network host, or its IPv4 or IPv6 address."
}

leaf locationProfilePort {
  type uint32;
  description
    "The port number with which to connect. Most schemes designate protocols that have a
default port number. If this is set to NULL, then the default port number is to be used. Else another port number may be supplied here."
}

leaf locationProfilePath {
  type string;
  description
    "The remaining parts of the URL necessary to completely define the desired transaction."
}

-- ends the locationProfileEntry --
--- The Network Profile defines a set of
--- reusable network layer parameters to fully
--- define the transaction test ultimately
--- defined in the Test Control.

list networkProfileEntry {
  key "networkProfileIndex";
  config true;

  leaf locationProfileIndex {
    type uint32;
    description
    "Identifies the specific network layer
    parameters for the transaction tests
    ultimately defined in the Control Table.";
  }

  leaf dstAddr {
    type inet:ip-address;
    description
    "Identifies the destination address in
    the packet headers of the transaction
    request message.";
  }

  leaf srcAddr {
    type inet:ip-address;
    description
    "Identifies the source address in the
    packet headers of the transaction
    request message.";
  }

  leaf noFrag {
    type Boolean;
    description
    "Defines the 'No Fragmentation' header
    setting in the IP packet headers of the
    transaction request message.";
  }
}
leaf TOS {
    type uint8;
    description
    "Identifies the TOS field of the IPv4 or IPv6 packet headers of the transaction request message. The TOS field is eight bits in length and this integer is to be converted to an 8 bit binary to define the appropriate TOS Field setting."
}

leaf flowLabel {
    type uint16;
    description
    "Identifies the Flow Label field of the IPv6 packet headers of the transaction request message. The Flow Label field is 16 bits in length and this integer is to be converted to an 16 bit binary to define the appropriate Flow Label Field setting. In the event that the protocolType is set to 'IPv4', then this value is to be set to zero and is to be ignored in the creation of the IPv4 packets."
}

leaf protocolType {
    type inet:ip-address-type;
    description
    "Identifies the network protocol type for the network packets generated as part of the transaction request messages. The allowed values are 'IPv4' or 'IPv6'."
}

leaf looseSrcRoute {
    type string;
    description
    "Identifies the Loose Source Route header extension for the IP packets forming the"
transaction request message.

-- ends the networkProfileEntry --

-- The Metric Profile performance aspects of tests, including, e.g., frequency, metric, success criteria, etc.

list metricProfileEntry {
  key "metricProfileIndex";
  config true;

  leaf metricProfileIndex {
    type uint32;
    description
    "Identifies the specific metric profile for use in the definition of the transaction tests in the Control Table."
  }

  leaf spacing {
    type uint32;
    description
    "The number of seconds between executing subsequent transactions."
  }

  leaf number {
    type uint32;
    description
    "The number of transactions to be executed."
  }

  leaf metric {
    type enumeration;
    enum loss {

description
"Holds the indication of whether
the transaction was successful (1)
or failed (0).";
}
enum delay {
  description
  "Holds the number of milliseconds
  for the successful transaction
  or '0' if the transaction failed.";
}
enum throughput {
  description
  "Holds the measured throughput
  in units of bytes/millisecond for
  the transaction if successful
  or '0' if failed.";
}
default "loss";
description
"The metric tracked by this specific test.
These values are held on the rawResults
if the specific test indicates storage
of raw data values."
}

leaf target {
  type uint32;
description
"The performance target for each transaction
measurement. A measured transaction is deemed
successful if its measured 'metric' value
falls within the limits defined by this
'target'. E.g.,
  if 'metric = loss', then 'target' must
  equal '1' indicating success if response
  received.
  if 'metric = delay', then responses
  received within 'target' milliseconds
  are counted as successful.
  if 'metric = throughput', then responses
  received with throughputs greater than
  'target' are counted as successful.

The target value carries the
units defined by the 'metric', i.e.,
unitless if 'metric = loss',


milliseconds if 'metric = delay',

bytes/milliseconds if 'metric = throughput'.

The server counts the number of transaction measurements that are deemed successful. This count is compared against 'threshold' to determine overall success or failure of the test.

default "1";

leaf threshold {
  type uint32;
  description
  "The threshold value that determines the pass/fail status reported to the client by this server in the 'verifyStatus' notification."
}

-- ends the metricProfileEntry --

---------------------------------------------------
-- The Control Table defines the test sets. --
---------------------------------------------------

list controlTableEntry {
  key "controlTableIndex";
  config true;

  leaf controlTableIndex {
    type uint32;
    description
    "Identifies the specific control table row of the transaction test template to be
executed, which represents the verification test sets to be performed on the device as part of the verify operation.

}  

leaf locationProfileIndex {

type uint32;

description
 "The index from the locationProfileEntry indicating the URL for this test.";

}

leaf networkProfileIndex {

type uint32;

description
 "The index from the locationProfileEntry indicating the URL for this test.";

}

leaf metricProfileIndex {

type uint32;

description
 "The index from the locationProfileEntry indicating the URL for this test.";

}

leaf rawResultCollection {

type enumeration;

enum off {

description
 "Indicates that the server will not store the raw transaction measurement values of type indicated by metric.";

}

enum on {

}
Indicates that the server will store the raw transaction measurement values of type indicated by metric. Further, these raw measurement values will be passed to the client through 'verifyStatus' notification's 'extendedStatus' node.

cfg: true;
default: "off";

description "A switch to turn ON or OFF the raw data collection and notification."


Internet-Draft          NETCONF Transaction Test               July 2010

-- ends the controlTableEntry --

-- The Results Table contains
--    the results from the test. --

list resultsTableEntry {
    key "resultsTableIndex";
    cfg: true;

    leaf resultsTableIndex {
        type uint32;
        description "Identifies the specific Control Table row of the transaction test template to be executed, which represents the verification test sets performed on the device as part of the verify"
operation;";

}

leaf startTime {
    type yang:date-and-time;
    config false;
    description
    "The time the first transaction
    was sent for the previous test.
    This is set each time the test
    is initiated from a client. When this
    value is reset, the value of the
    'result' node is set to
    'indeterminant' and the value of the
    'received' node is set to zero."

}

leaf received {
    type uint32;
    config false;
    description
    "The number of successful
    transactions received during
    the previous test. This value
    is initialized to zero prior to
    the instantiation of the test
    and is incremented by one for
    each received transaction response
    message. This is set each time the
    test is initiated from a client."

}

leaf result {
    type enumeration {
        enum indeterminant{
            description
            "Set to 'indeterminant' upon
            the initiation of a test."
        }
    }
}
enum success{
    description
    "Set to 'success' if the number of successful transactions exceeded the 'threshold'.";
}

enum failure{
    description
    "Set to 'failure' if the number of successful transactions is less than or equal to the 'threshold'.";
}

config false;

leaf-list rawResults {
    description
    "Holds the raw metric value for each transaction successfully recorded as part of the specific test. The units used for these values conform to the units defined with the 'metric' measured."

    type uint32;
    config false;
    min-elements 1;
}

-- ends the Results Table --

Figure 3
Authors' Addresses

Robert G. Cole
U.S. Army CERDEC
328 Hopkins Road
Aberdeen Proving Ground, MD 21005
USA
Phone: +1.410.278.6779
Email: robert.g.cole@us.army.mil
URI: http://www.cs.jhu/~rgcole/

Dan Romascanu
Avaya
Atidim Technology Park, Bldg. #3
Tel Aviv 61131
Israel
Email: dromasca@avaya.com

Andy Bierman
InterWorking Labs
303 Potrero Street, Suite 52
Santa Cruz, CA 95060-2760
USA
Email: andyb@iwl.com