Abstract

The Network Configuration Protocol (NETCONF) and RESTCONF provide mechanisms to manipulate configuration datastores. NMDA introduces additional datastores for systems that support more advanced processing chains converting configuration to operational state. However, client applications are not able to be aware of common events in these additional datastores of the management system, such as a applied configuration state change in NETCONF server or RESTCONF server, that may impact management applications. This document defines a YANG module that allows a client to receive additional notifications for some common system events pertaining to the Network Management Datastore Architecture (NMDA) defined in [RFC8342].

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 August 29, 2019.

Copyright Notice

Copyright (c) 2019 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
1. Introduction

The Network Configuration Protocol (NETCONF) [RFC6241] and RESTCONF [RFC8040] provides mechanisms to manipulate configuration datastores. NMDA introduces additional datastores (e.g., <intended>, <operational>) for systems that support more advanced processing chains converting configuration to operational state. However, client applications are not able to be aware of common events in those additional datastores of the management system, e.g., there are many background activities (e.g., NMDA datastore consistency checking [I-D.ietf-netmod-nmda-diff]) that happen during the time that configuration is committed to <running> to the time that the configuration is actually applied to <operational>. It is possible that some configuration could not be applied to <operational> due to either validation issues, or missing resource, etc. There is a need for user to know the applying result of <intended> datastore and the reason why the configuration were not applied.

This document define a YANG module that allows a NMS client to receive additional notifications for some common system events pertaining to the Network Management Datastore Architecture (NMDA) defined in [RFC8342]. These notifications are designed to support the monitoring of the base system events within the server and not specific to any network management protocols such as NETCONF and RESTCONF.
The solution presented in this document is backwards compatible with [RFC6470].

1.1. Terminology

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.

The following terms are defined in [RFC8342] and are not redefined here:

- operational state datastore
- running configuration datastore
- intended configuration datastore

2. NMDA Base Notifications for applied intended configuration

2.1. Overview

The YANG module in NETCONF Base Notifications [RFC6470] specifies the following 5 event notifications for the 'NETCONF' stream to notify a client application that the NETCONF server state has changed:

- netconf-config-change
- netconf-capability-change
- netconf-session-start
- netconf-session-end
- netconf-confirmed-commit

These event notifications used within the 'NETCONF' stream are accessible to clients via the subscription mechanism described in [RFC5277].

This document introduces NMDA specific extension which allows a client to receive 2 notifications for additional common system events as follows:

apply-intended-start: Generated when a server with network management protocol support detects that configuration applied
from intended has started. A server MAY optionally generate this event for both NETCONF session and non-NETCONF management sessions. Note that configuration applying process of the learned configuration, system-provided configuration, and default values defined by data models is not relevant to the events defined in this document. The apply-intended-start starts at the time when configuration is successfully committed to <running>, i.e., the confirmed-commit procedure has been completed. In case two commits are detected, if the second commit is confirmed commit, the confirming commit and confirmed commit should be treated as the same commit, i.e., intended-apply-start should not be sent again for persistent confirmed commit. If the second commit is not confirmed commit, intended-apply-start for the second commit should not be sent until nmda-intended-applied for the first commit has been sent.

apply-intended-updated: Generated when a server with network management protocol support detects that all or a set of configurations are successfully applied or none of them are applied. Indicates the event and the current state of the intended configuration applying. In addition, NMDA datastore compare [I-D.ietf-netmod-nmda-diff] should be used to check which part of intended configuration data is applied or which part of intended configuration data is not applied. A server MAY report events for non-NETCONF management sessions (such as RESTCONF, gPRC), using the 'session-id' value of zero.

The following figure shows event notification sequence defined in this document.
These notification messages are accessible to clients via either the subscription mechanism described in [RFC5277] or dynamic subscription mechanism and configured subscription mechanism described in [I-D.ietf-netconf-netconf-event-notifications].

2.2. Data Model Design

The data model is defined in the ietf-nmda-notifications YANG module. Its structure is shown in the following figure. The notation syntax follows [RFC8340].
notifications:

```yaml
---n apply-intended-start {apply-start}?
  +---ro username             string
  +---ro session-id           session-id-or-zero-type
  +---ro app-tag?             tags:tag
  +---ro source-host?         inet:ip-address
  +---ro commit-persist-id?   string

---n apply-intended-updated
  +---ro username             string
  +---ro session-id           session-id-or-zero-type
  +---ro app-tag?             tags:tag
  +---ro source-host?         inet:ip-address
  +---ro commit-persist-id?   string
  +---ro datastore?           identityref
  +---n (filter-spec)?        
     +--:(subtree-filter)
        |  +---ro subtree-filter?  <anydata>
     +--:(xpath-filter)
        |  +---ro xpath-filter?    yang:xpath1.0 {nc:xpath}? +---ro apply-result?  enumeration
  +---ro fail-applied-target* [edit-id]
     +---ro edit-id           string
     +---ro operation         enumeration
     +---ro target?           ypatch:target-resource-offset
     +---ro value?            <anydata>
     +---n errors
        +---n error* []
           +---ro error-type  enumeration
           +---ro error-tag   string
           +---ro error-app-tag?  string
           +---ro error-path?  instance-identifier
           +---ro error-message?  string
           +---ro error-info?  <anydata>
```

The following are examples of a apply-intended-updated notification message:
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2017-06-16T16:30:59.137045+09:00</eventTime>
    <username>admin</username>
    <session-id>0</session-id>
    <app-tag>user:app1</app-tag>
    <source-host>10.251.93.83</source-host>
    <commit-persist>IQ,d4668</commit-persist>
    <apply-result>fail</apply-result>
    <datastore>operational</datastore>
    <fail-applied-taget>
      <edit-id>1</edit-id>
      <operation>merge</operation>
      <target>/ietf-interfaces:interfaces-state</target>
      <value>
        <interfaces-state xmlns="http://foo.com/ietf-interfaces">
          <interface>
            <name>eth0</name>
            <oper-status>down</oper-status>
          </interface>
        </interfaces-state>
      </value>
    </fail-applied-target>
    <fail-applied-taget>
      <edit-id>2</edit-id>
      <target>/ietf-system:system</target>
      <errors>
        <error-type>protocol</error-type>
        <error-tag>mis-resource</error-tag>
        <error-path xmlns:ops="https://example.com/ns/ietf-system">\</if:interfaces-state\</error-path>
        <error-message>refer to resources that are not available or otherwise not physically present.</error-message>
      </errors>
    </fail-applied-target>
  </apply-intended-updated>
</notification>

2.3. Relation with NMDA Datastore Compare

NMDA datastore compare [NMDA-DIFF] could be used to check which part of intended configuration data is applied or which part of intended configuration data is not applied. On the other hand, the event notification for apply-intended-start or netconf-confirmed-commit can be used to trigger NMDA datastore compare procedure to be started.
2.4. Definitions

This section presents the YANG module defined in this document. This module imports data types from the 'ietf-datastores' module defined in [RFC8342] and 'ietf-inet-types' module defined in [RFC6021].

<CODE BEGINS> file "ietf-nmda-notifications@2018-12-26.yang"
module ietf-nmda-notifications {
  yang-version 1.1;
  prefix ndn;
  import ietf-datastores {
    prefix ds;
  }
  import ietf-inet-types { prefix inet; }
  import ietf-module-tags { prefix tags; }
  import ietf-yang-types { prefix yang; }
  import ietf-yang-patch { prefix ypatch; }
  import ietf-netconf { prefix nc; }
  import ietf-restconf { prefix rc; }
  organization "IETF NETMOD (Network Modeling) Working Group";
  contact "WG Web:  <http://tools.ietf.org/wg/netmod/>
              WG List:  <mailto:netmod@ietf.org>
              WG Chair: Kent Watsen
              <mailto:kwatsen@juniper.net>
              Editor:  Qin Wu
              <mailto:bill.wu@huawei.com>
              Editor:  Rohit R Ranade
              <mailto:rohitrranade@huawei.com>";
  description "This module defines a YANG data model for use with the
              NETCONF and RESTCONF protocol that allows the client to
              receive additional common event notifications related to NMDA.

  Copyright (c) 2012 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)."
typedef session-id-or-zero-type {
  type uint32;
  description
  "NETCONF Session Id or Zero to indicate none";
}

feature error-info {
  description
  "This feature must also be enabled for that session if error-info can
  be advertised by the server. Otherwise, this feature must not be enabled.";
}

feature apply-start {
  description
  "This feature must also be enabled for that session if apply-intended-start can
  be supported by the server. Otherwise, this feature must not be enabled.";
}

grouping common-session-parms {
  description
  "Common session parameters to identify a management session.";

  leaf username {
    type string;
    mandatory true;
    description
    "Name of the user for the session.";
  }

  leaf session-id {
    type session-id-or-zero-type;
    mandatory true;
    description
    "Identifier of the session.
     A NETCONF session MUST be identified by a non-zero value.
     A non-NETCONF session MAY be identified by the value zero.";
  }
}
leaf app-tag {
    type tags:tag;
    description
    "The relationship between username and application tag
    can be one to many.";
}

leaf source-host {
    type inet:ip-address;
    description
    "Address of the remote host for the session.";
}

leaf commit-persist-id {
    type string;
    description
    "This parameter is used to identify each commit.
    The value of this parameter is a token that must be given
    in the 'persist-id' parameter of <commit> or
    <cancel-commit> operations in order to confirm or cancel
    the persistent confirmed commit.

    The token should be a random string.";

    reference "RFC 6241, Section 8.3.4.1";
}

notification apply-intended-start {
    if-feature apply-start;
    description
    " Generated when a server detects that applied configuration
    From intended has started. This event will be sent immediately
    upon any data is written to <running>. A server MAY generate
    this event for both NETCONF session and non-NETCONF management
    sessions.";

    uses common-session-parms;
} // notification apply-intended-start

notification apply-intended-updated {
    description
    "Generated when a server detects that a
    intended configuration applied event has occurred. Indicates
    the event and the current state of the intended data applying
    procedure in progress.";

    reference "RFC 8342, Section 5";

    uses common-session-parms;

    leaf datastore {

type identityref {
  base ds:datapath;
}
default "ds:operational";
description
  "Indicates which datastore has changed or which datastore is
target of edit-data operation."
}
choice filter-spec {
  description
    "The content filter specification for this request."
  anydata subtree-filter {
    description
      "This parameter identifies the portions of the
target datastore to retrieve.";
    reference
      "RFC 6241: Network Configuration Protocol, Section 6.";
  }
  leaf xpath-filter {
    if-feature nc:xpath;
    type yang:xpath1.0;
    description
      "This parameter contains an XPath expression identifying
the portions of the target datastore to retrieve.

If the expression returns a node-set, all nodes in the
node-set are selected by the filter. Otherwise, if the
expression does not return a node-set, then the get-data
operation fails.

The expression is evaluated in the following XPath
context:

  o The set of namespace declarations are those in
    scope on the 'xpath-filter' leaf element.

  o The set of variable bindings is empty.

  o The function library is the core function library,
    and the XPath functions defined in section 10 in
    RFC 7950.

  o The context node is the root node of the target
datastore.";
  }
}
leaf apply-result {
  type enumeration {

enum "ok" {
  description
  "The intended data is succesfully applied."
}
enum "fail" {
  description
  "The intended data is not succesfully applied."
}

list fail-applied-target {
  when ".../apply-result = ‘fail’‘;
  key edit-id;
  ordered-by user;
  leaf edit-id {
    type string;
    description
    "Response status is for the ‘edit’ list entry with this ‘edit-id’ value."
  }
  leaf operation {
    type enumeration {
      enum create {
        description
        "The target data node is created using the supplied value, only if it does not already exist. The ‘target’ leaf identifies the data node to be created, not the parent data node."
      }
      enum delete {
        description
        "Delete the target node, only if the data resource currently exists; otherwise, return an error."
      }
      enum insert {
        description
        "Insert the supplied value into a user-ordered list or leaf-list entry. The target node must represent a new data resource. If the ‘where’ parameter is set to ‘before’ or ‘after’, then the ‘point’ parameter identifies the insertion point for the target node."
      }
      enum merge {
        description

"The supplied value is merged with the target data node."
}
enum move {
    description
    "Move the target node. Reorder a user-ordered list or leaf-list. The target node must represent an existing data resource. If the 'where' parameter is set to 'before' or 'after', then the 'point' parameter identifies the insertion point to move the target node."
}
enum replace {
    description
    "The supplied value is used to replace the target data node."
}
enum remove {
    description
    "Delete the target node if it currently exists."
}
}
mandatory true;

description
    "The datastore operation requested for the associated 'edit' entry."
}
leaf target {
    type ypatch:target-resource-offset;
    description
    "Topmost node associated with the configuration change. A server SHOULD set this object to the node within the datastore that is being altered. A server MAY set this object to one of the ancestors of the actual node that was changed, or omit this object, if the exact node is not known."
}
anydata value {
    description
    "Value used for this edit operation. The anydata 'value' contains the target resource associated with the 'target' leaf.

For example, suppose the target node is a YANG container named foo:

    container foo {
        leaf a { type string; }
    }
leaf b { type int32; }

The 'value' node contains one instance of foo:

```xml
<value>
  <foo xmlns='example-foo-namespace'>
    <a>some value</a>
    <b>42</b>
  </foo>
</value>
```

uses rc:errors {if-feature error-info;}

description
  "List for fail applied targets. The fail applied targets
  are only applied when the intended data is not applied."

3. Security Considerations

The YANG module defined in this memo is designed to be accessed via the NETCONF protocol [RFC6241]. The lowest NETCONF layer is the secure transport layer and the mandatory-to-implement secure transport is SSH, defined in [RFC6242].

Some of the readable data nodes in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes. These are the subtrees and data nodes and their sensitivity/vulnerability:

/apply-intended-start:

Event type itself indicates that a server may start applying configuration from intended. It may be possible for an attacker to slow down intended validation or update intended independently of running by somehow taking advantage of configuration template transformation.

/apply-intended-updated:

Event type itself indicates that a server may be finished applying configuration from intended. This event could alert an attacker that a datastore may have been altered.
/apply-intended-updated/apply-result:

Indicates the specific applied intended applied event state change that occurred. A value of ‘ok’ probably indicates that intended data applying procedure has completed.

4. IANA Considerations

This document registers one XML namespace URN in the ‘IETF XML registry’, following the format defined in [RFC3688]:


Registrant Contact: The IESG.

XML: N/A, the requested URI is an XML namespace.

This document registers one module name in the ‘YANG Module Names’ registry, defined in [RFC7950]:

name: ietf-nmda-notifications

prefix: ndn


RFC: xxxx

5. Acknowledgements

Thanks to Juergen Schoenwaelder, Alex Clemm, Carey Timothy and Andy Berman to review this draft and Thank Xiaojian Ding provide important input to the initial version of this document.

6. Normative References

[I-D.ietf-netmod-nmda-diff]
Clemm, A., Qu, Y., Tantsura, J., and A. Bierman, "Comparison of NMDA datastores", draft-ietf-netmod-nmda-diff-00 (work in progress), October 2018.


Authors’ Addresses

Qin Wu
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: bill.wu@huawei.com