

Internet Engineering Task Force (IETF)
Request for Comments: 8526
Updates: 6241, 7950
Category: Standards Track
ISSN: 2070-1721

M. Bjorklund
Tail-f Systems
J. Schoenwaelder
Jacobs University
P. Shafer
Juniper Networks
K. Watsen
Watsen Networks
R. Wilton
Cisco Systems
March 2019

NETCONF Extensions to Support the
Network Management Datastore Architecture

Abstract

This document extends the Network Configuration Protocol (NETCONF) defined in RFC 6241 in order to support the Network Management Datastore Architecture (NMDA) defined in RFC 8342.

This document updates RFCs 6241 and 7950. The update to RFC 6241 adds new <get-data> and <edit-data> operations and augments existing <lock>, <unlock>, and <validate> operations. The update to RFC 7950 requires the usage of the YANG library (described in RFC 8525) by NETCONF servers implementing the NMDA.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at <https://www.rfc-editor.org/info/rfc8526>.

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 (<https://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. Terminology	3
1.2. Tree Diagrams	3
2. Datastore and YANG Library Requirements	3
3. NETCONF Extensions	4
3.1. New NETCONF Operations	4
3.1.1. The <get-data> Operation	4
3.1.2. The <edit-data> Operation	10
3.2. Augmentations to NETCONF Operations	11
4. NETCONF Datastores YANG Module	12
5. IANA Considerations	20
6. Security Considerations	21
7. References	21
7.1. Normative References	21
7.2. Informative References	22
Authors' Addresses	23

1. Introduction

This document extends the NETCONF protocol defined in [RFC6241] in order to support the Network Management Datastore Architecture (NMDA) defined in [RFC8342].

This document updates [RFC6241] in order to enable NETCONF clients to interact with all the datastores supported by a server implementing the NMDA. The update both adds new <get-data> and <edit-data> operations and augments existing <lock>, <unlock>, and <validate> operations.

This document also updates [RFC7950] in order to enable NETCONF clients to both discover which datastores are supported by the NETCONF server and determine which modules are supported in each datastore. The update requires NETCONF servers implementing the NMDA to support the YANG library [RFC8525].

1.1. Terminology

This document uses the terminology defined by the NMDA [RFC8342].

The following term is defined in [RFC8525]:

- o YANG library content identifier

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.

1.2. Tree Diagrams

Tree diagrams used in this document follow the notation defined in [RFC8340].

2. Datastore and YANG Library Requirements

An NMDA-compliant NETCONF server MUST implement the "ietf-netconf-nmda" module defined in this document, MUST support the operational state datastore, and MUST implement at least revision 2019-01-04 of the "ietf-yang-library" module defined in [RFC8525].

A NETCONF client can discover which datastores and YANG modules the server supports by reading the YANG library information from the operational state datastore.

The server MUST advertise the following capability in the <hello> message (line breaks and whitespace are used for formatting reasons only):

```
urn:ietf:params:netconf:capability:yang-library:1.1?
  revision=<date>&content-id=<content-id-value>
```

The parameter "revision" has the same value as the revision date of the "ietf-yang-library" module implemented by the server. This parameter MUST be present.

The parameter "content-id" contains the YANG library content identifier [RFC8525]. This parameter MUST be present.

With this mechanism, a client can cache the supported datastores and YANG modules for a server and only update the cache if the "content-id" value in the <hello> message changes.

This document updates Section 5.6.4 of [RFC7950] to allow servers to advertise the capability :yang-library:1.1 instead of :yang-library:1.0 and to implement the subtree "/yang-library" [RFC8525] instead of "/modules-state".

3. NETCONF Extensions

This section describes the NETCONF extensions needed to support the NMDA. These changes are defined in the new "ietf-netconf-nmda" YANG [RFC7950] module.

These changes include the use of source and target parameters based on the "datastore" identity defined in the "ietf-datastores" module [RFC8342]. The use of identities allows future expansion in a way that the choice-based strategy from the original operations (e.g., <get-config> and <edit-config>) does not.

3.1. New NETCONF Operations

Two new operations -- <get-data> and <edit-data> -- are defined in this document in order to support the NMDA. These operations are similar to the <get-config> and <edit-config> operations, but they can work on an extensible set of datastores.

3.1.1. The <get-data> Operation

The <get-data> operation retrieves data from a specific NMDA datastore. This operation is similar to NETCONF's <get-config> operation defined in [RFC6241], but it adds the flexibility to select the source datastore.

```

+---x get-data
  +---w input
    +---w datastore                               ds:datastore-ref
    +---w (filter-spec)?
      +---:(subtree-filter)
      |   +---w subtree-filter?                 <anydata>
      |   +---:(xpath-filter)
      |       +---w xpath-filter?             yang:xpath1.0 {nc:xpath}?
    +---w config-filter?                          boolean
    +---w (origin-filters)? {origin}?
      +---:(origin-filter)
      |   +---w origin-filter*                 or:origin-ref
      |   +---:(negated-origin-filter)
      |       +---w negated-origin-filter*    or:origin-ref
    +---w max-depth?                             union
    +---w with-origin?                           empty {origin}?
    +---w with-defaults?                       with-defaults-mode
  +--ro output
    +--ro data?   <anydata>

```

The "datastore" parameter indicates the datastore that is the source of the data to be retrieved. This is a "datastore" identity.

The <get-data> operation accepts a content filter parameter, similar to the "filter" parameter of <get-config>, but uses explicit nodes for subtree filtering ("subtree-filter") and XPath filtering ("xpath-filter").

The "config-filter" parameter can be used to retrieve only "config true" or "config false" nodes.

The "origin-filter" parameter, which can be present multiple times, selects nodes equal to or derived from any of the given values. The "negated-origin-filter", which can be present multiple times, selects nodes that are not equal to or derived from any of the given values. The "origin-filter" and "negated-origin-filter" parameters cannot be used together.

The "max-depth" parameter can be used by the client to limit the number of subtree levels that are returned in the reply.

3.1.1.1. "origin" Metadata Annotation

The <get-data> operation defines a parameter named "with-origin", which if present, requests that the server includes "origin" metadata annotations in its response, as detailed in the NMDA. This parameter is only valid for the operational state datastore and any datastores with identities derived from the "operational" identity. Otherwise,

if an invalid datastore is specified then an error is returned, as specified in the "ietf-netconf-nmda" module (see Section 4). Note that "origin" metadata annotations are not included in a response unless a client explicitly requests them.

Data in the operational state datastore can come from multiple sources. The server should return the "origin" metadata annotation value that most accurately indicates the source of the operational value, as specified in Section 5.3.4 of [RFC8342].

When encoding the "origin" metadata annotation for a hierarchy of returned nodes, the annotation may be omitted for a child node when the value matches that of the parent node, as described in the "ietf-origin" YANG module [RFC8342].

Support for the "with-origin" parameter is OPTIONAL. It is identified with the feature "origin".

3.1.1.2. "with-defaults" Interactions

If the "with-defaults" capability is supported by the server, then the "with-defaults" parameter, defined in [RFC6243], is supported for <get-data> operations that target conventional configuration datastores.

Support for the "with-defaults" parameter is OPTIONAL for <get-data> operations that target <operational>. The associated capability to indicate a server's support is identified with the URI:

urn:ietf:params:netconf:capability:with-operational-defaults:1.0

If the "with-defaults" parameter is supported for <get-data> operations on <operational>, then all retrieval modes specified in either the 'basic-mode' or 'also-supported' parameter of the "with-defaults" capability are permitted. The behavior of the "with-defaults" parameter for <operational> is defined as below:

- o If no "with-defaults" parameter is specified, or if it is set to "explicit", "report-all", or "report-all-tagged", then the "in use" values, as defined in Section 5.3 of [RFC8342], are returned from the operational state datastore, even if a node happens to have a default statement in the YANG module, and this default value is being used by the server. If the "with-defaults" parameter is set to "report-all-tagged", any values that match the schema default are tagged with additional metadata, as described in Section 3.4 of [RFC6243].

- o If the "with-defaults" parameter is set to "trim", all "in use" values are returned, except that the output is filtered to exclude any values that match the default defined in the YANG schema.

Support for "with-defaults" in <get-data> operations on any datastore not defined in [RFC8342] should be defined by the specification for the datastore.

3.1.1.3. Example: Retrieving an Entire Subtree from <running>

The following example shows the <get-data> version of the <get-config> example shown in Section 7.1 of [RFC6241], which selects the entire "/users" subtree:

```
<rpc message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-data xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-nmda"
    xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">
    <datastore>ds:running</datastore>
    <subtree-filter>
      <top xmlns="http://example.com/schema/1.2/config">
        <users/>
      </top>
    </subtree-filter>
  </get-data>
</rpc>

<rpc-reply message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-nmda">
    <top xmlns="http://example.com/schema/1.2/config">
      <users>
        <user>
          <name>root</name>
          <type>superuser</type>
          <full-name>Charlie Root</full-name>
          <company-info>
            <dept>1</dept>
            <id>1</id>
          </company-info>
        </user>
        <!-- additional <user> elements appear here... -->
      </users>
    </top>
  </data>
</rpc-reply>
```

3.1.1.4. Example: Retrieving a Filtered Subtree from <operational>

The following example shows how the "origin-filter" can be used to retrieve nodes from <operational>. The example uses the fictional data model defined in Appendix C of [RFC8342].

```
<rpc message-id="102"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-data xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-nmda"
    xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores"
    xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin">
    <datastore>ds:operational</datastore>
    <subtree-filter>
      <bgp xmlns="http://example.com/ns/bgp"/>
    </subtree-filter>
    <origin-filter>or:intended</origin-filter>
    <origin-filter>or:system</origin-filter>
    <with-origin/>
  </get-data>
</rpc>

<rpc-reply message-id="102"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-nmda">
    <bgp xmlns="http://example.com/ns/bgp"
      xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin"
      or:origin="or:intended">
      <peer>
        <name>2001:db8::2:3</name>
        <local-port or:origin="or:system">60794</local-port>
        <state>established</state>
      </peer>
    </bgp>
  </data>
</rpc-reply>
```


In order to not retrieve any system state nodes, the "config-filter" can be used:

```
<rpc message-id="103"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-data xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-nmda"
    xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores"
    xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin">
    <datastore>ds:operational</datastore>
    <subtree-filter>
      <bgp xmlns="http://example.com/ns/bgp"/>
    </subtree-filter>
    <config-filter>true</config-filter>
    <origin-filter>or:intended</origin-filter>
    <origin-filter>or:system</origin-filter>
    <with-origin/>
  </get-data>
</rpc>

<rpc-reply message-id="103"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-nmda">
    <bgp xmlns="http://example.com/ns/bgp"
      xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin"
      or:origin="or:intended">
      <peer>
        <name>2001:db8::2:3</name>
        <local-port or:origin="or:system">60794</local-port>
      </peer>
    </bgp>
  </data>
</rpc-reply>
```

3.1.2. The <edit-data> Operation

The <edit-data> operation changes the contents of a writable datastore, similar to the <edit-config> operation defined in [RFC6241] but with additional flexibility in naming the target datastore. If an <edit-data> operation is invoked on a non-writable datastore, then an error is returned, as specified in the "ietf-netconf-nmda" module (see Section 4).

```
+---x edit-data
  +---w input
    +---w datastore           ds:datastore-ref
    +---w default-operation?  enumeration
    +---w (edit-content)
      +--:(config)
      | +---w config?         <anydata>
      +--:(url)
        +---w url?           inet:uri {nc:url}?
```

The "datastore" parameter is a "datastore" identity that indicates the desired target datastore where changes should be made.

The "default-operation" parameter selects the default operation to use. It is a copy of the "default-operation" parameter of the <edit-config> operation.

The "edit-content" parameter specifies the content for the edit operation. It mirrors the "edit-content" choice of the <edit-config> operation. Note, however, that the "config" element in the "edit-content" choice of <edit-data> uses "anydata" (introduced in YANG 1.1 [RFC7950]) while the "config" element in the "edit-content" choice of <edit-config> used "anyxml".

The <edit-data> operation does not support the "error-option" and the "test-option" parameters that were part of the <edit-config> operation. The error behavior of <edit-data> corresponds to the "rollback-on-error" value in the "error-option" parameter.

If the "with-defaults" capability is supported by the server, the semantics of editing modes is the same as for <edit-config>, as described in Section 4.5.2 of [RFC6243].

Semantics for "with-defaults" in <edit-data> operations on any non-conventional configuration datastores should be defined by the specification for the datastore.

3.1.2.1. Example: Setting a Leaf of an Interface in <running>

The following example shows the <edit-data> version of the first <edit-config> example in Section 7.2 of [RFC6241]. In this example, the MTU is set to 1500 on an interface named "Ethernet0/0" in the running configuration datastore.

```
<rpc message-id="103"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-data xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-nmda"
    xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">
    <datastore>ds:running</datastore>
    <config>
      <top xmlns="http://example.com/schema/1.2/config">
        <interface>
          <name>Ethernet0/0</name>
          <mtu>1500</mtu>
        </interface>
      </top>
    </config>
  </edit-data>
</rpc>

<rpc-reply message-id="103"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ok/>
</rpc-reply>
```

The other <edit-config> examples shown in Section 7.2 of [RFC6241] can be translated to <edit-data> examples in a similar way.

3.2. Augmentations to NETCONF Operations

Several of the operations defined in the base NETCONF YANG module "ietf-netconf" [RFC6241] may be used with new datastores. Hence, the <lock>, <unlock>, and <validate> operations are augmented with a new "datastore" leaf that can select the desired datastore. If a <lock>, <unlock>, or <validate> operation is not supported on a particular datastore, then an error is returned, as specified in the "ietf-netconf-nmda" module (see Section 4).

4. NETCONF Datastores YANG Module

This module imports definitions from [RFC6991], [RFC6241], [RFC6243], and [RFC8342].

```
<CODE BEGINS> file "ietf-netconf-nmda@2019-01-07.yang"

module ietf-netconf-nmda {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-netconf-nmda";
  prefix ncds;

  import ietf-yang-types {
    prefix yang;
    reference
      "RFC 6991: Common YANG Data Types";
  }
  import ietf-inet-types {
    prefix inet;
    reference
      "RFC 6991: Common YANG Data Types";
  }
  import ietf-datastores {
    prefix ds;
    reference
      "RFC 8342: Network Management Datastore Architecture
      (NMDA)";
  }
  import ietf-origin {
    prefix or;
    reference
      "RFC 8342: Network Management Datastore Architecture
      (NMDA)";
  }
  import ietf-netconf {
    prefix nc;
    reference
      "RFC 6241: Network Configuration Protocol (NETCONF)";
  }
  import ietf-netconf-with-defaults {
    prefix ncwd;
    reference
      "RFC 6243: With-defaults Capability for NETCONF";
  }
}

organization
  "IETF NETCONF Working Group";
```

contact

"WG Web: <<https://datatracker.ietf.org/wg/netconf/>>

WG List: <<mailto:netconf@ietf.org>>

Author: Martin Bjorklund
<<mailto:mbj@tail-f.com>>

Author: Juergen Schoenwaelder
<<mailto:j.schoenwaelder@jacobs-university.de>>

Author: Phil Shafer
<<mailto:phil@juniper.net>>

Author: Kent Watsen
<<mailto:kent+ietf@watsen.net>>

Author: Robert Wilton
<<mailto:rwilton@cisco.com>>";

description

"This YANG module defines a set of NETCONF operations to support the Network Management Datastore Architecture (NMDA).

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 (RFC 2119) (RFC 8174) when, and only when, they appear in all capitals, as shown here.

Copyright (c) 2019 IETF Trust and the persons identified as authors of the code. 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 (<https://trustee.ietf.org/license-info>).

This version of this YANG module is part of RFC 8526; see the RFC itself for full legal notices.";

revision 2019-01-07 {

description

"Initial revision.";

reference

"RFC 8526: NETCONF Extensions to Support the Network Management Datastore Architecture";

```
}  
  
feature origin {  
  description  
    "Indicates that the server supports the 'origin' annotation.";  
  reference  
    "RFC 8342: Network Management Datastore Architecture (NMDA)";  
}  
  
feature with-defaults {  
  description  
    "NETCONF :with-defaults capability. If the server advertises  
    the :with-defaults capability for a session, then this  
    feature must also be enabled for that session. Otherwise,  
    this feature must not be enabled.";  
  reference  
    "RFC 6243: With-defaults Capability for NETCONF, Section 4; and  
    RFC 8526: NETCONF Extensions to Support the Network Management  
    Datastore Architecture, Section 3.1.1.2";  
}  
  
rpc get-data {  
  description  
    "Retrieve data from an NMDA datastore. The content returned  
    by get-data must satisfy all filters, i.e., the filter  
    criteria are logically ANDed.  
  
    Any ancestor nodes (including list keys) of nodes selected by  
    the filters are included in the response.  
  
    The 'with-origin' parameter is only valid for an operational  
    datastore. If 'with-origin' is used with an invalid  
    datastore, then the server MUST return an <rpc-error> element  
    with an <error-tag> value of 'invalid-value'.  
  
    The 'with-defaults' parameter only applies to the operational  
    datastore if the NETCONF :with-defaults and  
    :with-operational-defaults capabilities are both advertised.  
    If the 'with-defaults' parameter is present in a request for  
    which it is not supported, then the server MUST return an  
    <rpc-error> element with an <error-tag> value of  
    'invalid-value'.";  
  input {  
    leaf datastore {  
      type ds:datastore-ref;  
      mandatory true;  
    }  
  }  
}
```

```

description
  "Datastore from which to retrieve data.

  If the datastore is not supported by the server, then the
  server MUST return an <rpc-error> element with an
  <error-tag> value of 'invalid-value'.";
}
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 (NETCONF),
      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 are defined in Section 10
        of RFC 7950.

      o The context node is the root node of the target
        datastore.";
  }
}
leaf config-filter {
  type boolean;

```

```

description
  "Filter for nodes with the given value for their 'config'
  property. When this leaf is set to 'true', only 'config
  true' nodes are selected, and when set to 'false', only
  'config false' nodes are selected. If this leaf is not
  present, no nodes are filtered.";
}
choice origin-filters {
  when 'derived-from-or-self(datastore, "ds:operational")';
  if-feature "origin";
  description
    "Filters configuration nodes based on the 'origin'
    annotation. Configuration nodes that do not have an
    'origin' annotation are treated as if they have the
    'origin' annotation 'or:unknown'."

    System state nodes are not affected by origin-filters and
    thus not filtered. Note that system state nodes can be
    filtered with the 'config-filter' leaf.";

  leaf-list origin-filter {
    type or:origin-ref;
    description
      "Filter based on the 'origin' annotation. A
      configuration node matches the filter if its 'origin'
      annotation is derived from or equal to any of the given
      filter values.";
  }
  leaf-list negated-origin-filter {
    type or:origin-ref;
    description
      "Filter based on the 'origin' annotation. A
      configuration node matches the filter if its 'origin'
      annotation is neither derived from nor equal to any of
      the given filter values.";
  }
}
leaf max-depth {
  type union {
    type uint16 {
      range "1..65535";
    }
    type enumeration {
      enum unbounded {
        description
          "All descendant nodes are included.";
      }
    }
  }
}

```



```

    }
    default "unbounded";
    description
      "For each node selected by the filters, this parameter
       selects how many conceptual subtree levels should be
       returned in the reply.  If the depth is 1, the reply
       includes just the selected nodes but no children.  If the
       depth is 'unbounded', all descendant nodes are included.";
  }
  leaf with-origin {
    when 'derived-from-or-self(../datastore, "ds:operational)';
    if-feature "origin";
    type empty;
    description
      "If this parameter is present, the server will return
       the 'origin' annotation for the nodes that have one.";
  }
  uses ncwd:with-defaults-parameters {
    if-feature "with-defaults";
  }
}
output {
  anydata data {
    description
      "Copy of the source datastore subset that matched
       the filter criteria (if any).  An empty data
       container indicates that the request did not
       produce any results.";
  }
}
}

rpc edit-data {
  description
    "Edit data in an NMDA datastore.

    If an error condition occurs such that an error severity
    <rpc-error> element is generated, the server will stop
    processing the <edit-data> operation and restore the
    specified configuration to its complete state at
    the start of this <edit-data> operation.";
  input {
    leaf datastore {
      type ds:datastore-ref;
      mandatory true;
    }
  }
}

```

```

description
  "Datastore that is the target of the <edit-data> operation.

  If the target datastore is not writable, or is not
  supported by the server, then the server MUST return an
  <rpc-error> element with an <error-tag> value of
  'invalid-value'.";
}
leaf default-operation {
  type enumeration {
    enum merge {
      description
        "The default operation is merge.";
    }
    enum replace {
      description
        "The default operation is replace.";
    }
    enum none {
      description
        "There is no default operation.";
    }
  }
  default "merge";
  description
    "The default operation to use.";
}
choice edit-content {
  mandatory true;
  description
    "The content for the edit operation.";
  anydata config {
    description
      "Inline config content.";
  }
  leaf url {
    if-feature "nc:url";
    type inet:uri;
    description
      "URL-based config content.";
  }
}
}
}
/*
 * Augment the <lock> and <unlock> operations with a
 * "datastore" parameter.

```

```

*/
augment "/nc:lock/nc:input/nc:target/nc:config-target" {
  description
    "Add NMDA datastore as target.";
  leaf datastore {
    type ds:datastore-ref;
    description
      "Datastore to lock.

      The <lock> operation is only supported on writable
      datastores.

      If the <lock> operation is not supported by the server on
      the specified target datastore, then the server MUST return
      an <rpc-error> element with an <error-tag> value of
      'invalid-value'.";
  }
}

augment "/nc:unlock/nc:input/nc:target/nc:config-target" {
  description
    "Add NMDA datastore as target.";
  leaf datastore {
    type ds:datastore-ref;
    description
      "Datastore to unlock.

      The <unlock> operation is only supported on writable
      datastores.

      If the <unlock> operation is not supported by the server on
      the specified target datastore, then the server MUST return
      an <rpc-error> element with an <error-tag> value of
      'invalid-value'.";
  }
}

/*
 * Augment the <validate> operation with a
 * "datastore" parameter.
 */

augment "/nc:validate/nc:input/nc:source/nc:config-source" {
  description
    "Add NMDA datastore as source.";
  leaf datastore {
    type ds:datastore-ref;

```

```
description
  "Datastore to validate.
```

```

  The <validate> operation is supported only on configuration
  datastores.
```

```

  If the <validate> operation is not supported by the server
  on the specified target datastore, then the server MUST
  return an <rpc-error> element with an <error-tag> value of
  'invalid-value'.";
```

```

}
}
}
```

```
<CODE ENDS>
```

5. IANA Considerations

This document registers two capability identifier URNs in the "Network Configuration Protocol (NETCONF) Capability URNs" registry:

```
Index
Capability Identifier
```

```
-----
:yang-library:1.1
urn:ietf:params:netconf:capability:yang-library:1.1
```

```
:with-operational-defaults
urn:ietf:params:netconf:capability:with-operational-defaults:1.0
```

This document registers a URI in the "IETF XML Registry" [RFC3688]. Following the format in RFC 3688, the following registration has been made.

```
URI: urn:ietf:params:xml:ns:yang:ietf-netconf-nmda
```

```
Registrant Contact: The IESG.
```

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

This document registers a YANG module in the "YANG Module Names" registry [RFC6020].

```
name:          ietf-netconf-nmda
namespace:     urn:ietf:params:xml:ns:yang:ietf-netconf-nmda
prefix:        ncds
reference:     RFC 8526
```

6. Security Considerations

The YANG module defined in this document extends the base operations of the NETCONF [RFC6241] protocol. The lowest NETCONF layer is the secure transport layer and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242].

The Network Configuration Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular NETCONF users to a preconfigured subset of all available NETCONF protocol operations and content.

The security considerations for the base NETCONF protocol operations (see Section 9 of [RFC6241]) apply to the new NETCONF <get-data> and <edit-data> operations defined in this document.

7. References

7.1. 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>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <<https://www.rfc-editor.org/info/rfc6020>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <<https://www.rfc-editor.org/info/rfc6242>>.
- [RFC6243] Bierman, A. and B. Lengyel, "With-defaults Capability for NETCONF", RFC 6243, DOI 10.17487/RFC6243, June 2011, <<https://www.rfc-editor.org/info/rfc6243>>.

- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <<https://www.rfc-editor.org/info/rfc6991>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.
- [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>>.
- [RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <<https://www.rfc-editor.org/info/rfc8341>>.
- [RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <<https://www.rfc-editor.org/info/rfc8342>>.
- [RFC8525] Bierman, A., Bjorklund, M., Schoenwaelder, J., Watsen, K., and R. Wilton, "YANG Library", RFC 8525, DOI 10.17487/RFC8525, March 2019, <<https://www.rfc-editor.org/info/rfc8525>>.

7.2. Informative References

- [RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <<https://www.rfc-editor.org/info/rfc8340>>.

Authors' Addresses

Martin Bjorklund
Tail-f Systems

Email: mbj@tail-f.com

Juergen Schoenwaelder
Jacobs University

Email: j.schoenwaelder@jacobs-university.de

Phil Shafer
Juniper Networks

Email: phil@juniper.net

Kent Watsen
Watsen Networks

Email: kent+ietf@watsen.net

Robert Wilton
Cisco Systems

Email: rwilton@cisco.com

