Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with Section 6 of BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on May 7, 2009.

Copyright Notice

Copyright (C) The IETF Trust (2008).
Abstract

This document defines NETCONF content via XML Schema to be used to monitor the NETCONF protocol. The monitoring data model includes information about NETCONF sessions, locks, and subscriptions and is intended to facilitate management of a NETCONF server. In addition this monitoring data provides clients with standardized content to describe supported schema.

Today, NETCONF capabilities exchange is the only standardized method a client can use to discover the functionality supported by a NETCONF server. This works well for static protocol capabilities but is not well suited for capabilities which could change during a session.

Considerations such as different schema formats, feature optionality and access controls can all impact the applicability and level of detail the NETCONF server sends to a client during session setup. Through updated monitoring data NETCONF clients can adjust their capabilities throughout a session. Specifically the details returned can be used by a client to determine whether retrieval of new schema information is required and includes the information required to facilitate the retrieval.

A new RPC (get-schema) is also defined to support explicit schema retrieval via NETCONF.
Table of Contents

1. Introduction ............................................. 4
  1.1. Definition of Terms .................................... 4
2. XML Schema to Monitor NETCONF .............................. 5
  2.1. netconf ................................................. 5
    2.1.1. capabilities ........................................ 6
    2.1.2. configurations ...................................... 6
    2.1.3. schemas ............................................. 6
    2.1.4. sessions ............................................ 7
    2.1.5. subscriptions ...................................... 8
    2.1.6. statistics .......................................... 9
3. Schema Specific Operations ................................. 11
  3.1. New NETCONF RPC, <get-schema> .......................... 11
  3.2. NETCONF Schema List Retrieval (<get> monitoring data) .. 11
4. Examples .................................................... 13
  4.1. Retrieving schema list via <get> ....................... 13
  4.2. Retrieving schema instances ............................ 15
5. XSD Schema .................................................. 17
  5.1. NETCONF Monitoring Schema .............................. 17
  5.2. inet:host schema ........................................ 28
6. Security Considerations .................................... 31
7. Acknowledgements .......................................... 32
8. IANA Considerations ....................................... 33
9. Normative References ...................................... 34
Appendix A. YANG module (non-normative) ....................... 35
Authors’ Addresses ........................................... 43
Intellectual Property and Copyright Statements ................ 44
1. Introduction

This document defines NETCONF content via [XMLSchema] to be used to monitor the NETCONF protocol. It provides information about NETCONF sessions and subscriptions.

1.1. Definition of Terms

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 [RFC2119].

Operation: This term is used to refer to NETCONF protocol operations. Specifically within this document, operation refers to NETCONF protocol operations defined in support of NETCONF monitoring.

Schema: This term is used to refer to a data model fragment, independent of which data modeling language is used in the data model.

Subscription: A concept related to the delivery of notifications (if any to send) involving destination and selection of notifications. It is bound to the lifetime of a session.


YANG: YANG is a data modeling language used to model configuration and state data manipulated by the NETCONF protocol, NETCONF remote procedure calls, and NETCONF notifications.
2. XML Schema to Monitor NETCONF

The following data allows a NETCONF client to monitor both the
NETCONF server itself and the associated network device operational
data. The specific monitoring data defined in this draft which MUST
be present follows.

2.1. netconf

Root of the monitoring data model. Acts as container for the other
monitored data.

```
netconf
  |_capabilities
  |_configurations
  |_schemas
  |_sessions
  |_subscriptions
  |_statistics
```

capabilities (type: xs:anyURI)
List of NETCONF capabilities supported by the server.

configurations (type: ConfigurationDatastore)
List of NETCONF datastores on the server.
Includes all supported datastore types (running, candidate, startup)

schemas (type: SchemaEntry)
List of schemas supported on the server.
Includes all the information required to identify the schema and
to support its retrieval.

sessions (type: ManagementSession)
List of all active sessions on the device including NETCONF
and other sessions (eg: CLI).

subscriptions (type: NetconfSubscription)
List of all NETCONF notification subscriptions active on the server.

statistics (type: ManagementStatistics)
Contains management stats for the NETCONF server
including performance and error related counters.
2.1.1. capabilities

The list of capabilities supported by the NETCONF server. The list MUST include all capabilities exchanged during session setup still applicable at the time of the request. This ensures consistency with the initial capabilities exchanged while allowing for potential modifications during a session.

2.1.2. configurations

Configuration data pertaining to the NETCONF server.

configuration (type: ConfigurationDataStore)
  |_name
  |_locks

name (type: NETCONFDatastoreType)
  Enumeration of supported datastores; candidate, running, startup.

locks (type: ManagementSessionLock)
  Contains current lock information for the datastores. Global and partial locks are supported as either a single global lock or multiple partial locks per datastore.

Both a global lock and a partial lock MUST contain the sessionId and time the lock was acquired. The lockedNodes is the list of actual nodes which were locked at the the selection was evaluated. This is provided for completeness noting that the list of nodes returned by the selection may differ over time.

Additionally partial lock items will contain both the selection and the list of lockedNodes. The selection is the xpath expression which was used to evaluate the list of nodes to lock.

2.1.3. schemas
The list of supported schema for the NETCONF server.

```
schema
    _identifier   (key)
    _version      (key)
    _format       (key)
    _namespace
    _location
```

The elements identifier, version, and format are used as a key in the schema list. They are also used in the <get-schema> RPC.

**identifier (type: xs:string)**
Identifier for the schema list entry. Typically the module name or filename of the schema. It is also used in the <get-schema> RPC and may be used for other means.

**version (type: xs:string)**
Version of the schema supported. Multiple versions MAY be supported simultaneously by a NETCONF server. Each MUST be reported individually in the schema list, i.e. with same identifier, possibly different location, but different version.

**format (type: SchemaFormat)**
The data modeling language of the file/module. Current selection of XSD, YANG, and RNG.

**namespace (type: xs:anyURI)**
The XML namespace defined by the data model.

**location (union type: xs:string, xs:anyURI)**
A location from which this specific schema, format, and version can be retrieved.

At least one schema entry SHOULD be available to support retrieval.

A schema entry may be located on a network device (eg: xs:anyURI), (eg: xs:string reference to file system for ftp retrieval) or available explicitly via NETCONF (xs:string value ‘NETCONF’) for NETCONF servers which support the schema-retrieval capability via <get-schema>.

2.1.4. sessions
Session data pertaining to the NETCONF server. Includes data for NETCONF and non-NETCONF management sessions.

sessions (type: ManagementSessionInfo):

session
  |_sessionId
  |_transport
  |_protocol
  |_username
  |_sourceHost
  |_loginTime

sessionId (type: SessionId)
  Unique identifier for the session.

transport (type: TransportType)
  Identities transport for each session, e.g. "Console", "SSH".

protocol (type: ProtocolType)
  Identifies the protocol being used for each session, e.g.:
  "NETCONF", "CLI", "WebUI".

username (type: xs:string)
  Session owner.

sourceHost (type: inet:host)
  Host identifier (IP + name) for the client.
  See section 5.2 for definition.

loginTime (type: xs:dateTime)
  Time at which the session was established.

2.1.5. subscriptions
subscriptions: NetconfSubscription complex type.

List of notifications subscriptions for all active sessions.

subscription
   |_sessionId
   |_stream
   |_filter
   |_startTime
   |_stopTime
   |_messagesSent

sessionId (type: SessionId)
   Unique identifier for the notifications.
   MUST return the same value as returned in ‘sessions’ to allow correlation.

stream (netconf notification type: streamNameType)
   Specific event notifications subscribed by session.
   Unique identifier mappable to event subscriptions.

filter (netconf notification type: FilterType)
   Filter being applied to the notification stream.

startTime (xs:dateTime)
   Optional startTime parameter given in the <create-subscription>
   request.

stopTime (xs:dateTime)
   Optional stopTime parameter given in the <create-subscription>
   request.

messagesSent (xs:unsignedInt)
   Total messages sent for this subscription.

2.1.6. statistics
Statistical data pertaining to the NETCONF server.

managementStatistics(type: ManagementPerformanceStatistics):
  Contains management session related performance data for the NETCONF server.

netconfStartTime (type: xs:dateTime)
  Date and time at which the NETCONF server process was started.
  Allows for calculation of simple time interval for reported metrics.
  I.e: current time - startTime defines the collection interval for the metrics allowing for calculations such as averages.
  More complex calculations would require multiple collection intervals with both start and stop times defined per interval.

inSessions (type: xs:unsignedInt)
  The total number of NETCONF sessions started towards the NETCONF peer.

inXMLParseErrors (type: xs:unsignedInt)
  The total number of messages that were unparsable and thus ignored.
  This covers both unparsable <hello> and <rpc> messages.

inBadHellos (type: xs:unsignedInt)
  The total number of sessions silently dropped because an invalid <hello> message was received.
  This includes <hello> messages with a "session-id" attribute, bad namespace, and bad capability declarations.

inRpcs (type: xs:unsignedInt)
  The total number of rpc requests received.

inBadRpcs (type: xs:unsignedInt)
  The total number of rpcs which were parsed correctly, but couldn’t be serviced because they contained non-conformant XML, e.g. missing a mandatory parameter.

inNotSupportedRpcs (type: xs:unsignedInt)
  The total number of rpcs which were parsed correctly, but couldn’t be serviced because they were not supported by the agent.

outRpcReplies (type: xs:unsignedInt)
  The total number of <rpc-reply> messages sent.

outRpcErrors (type: xs:unsignedInt)
  The total number of <rpc-reply> messages with <rpc-error> sent.

outNotifications (type: xs:unsignedInt)
  The total number of <notification> messages sent.
3. Schema Specific Operations

3.1. New NETCONF RPC, <get-schema>

Description: When the schema is available on the device and the client wishes to have it returned via NETCONF this new RPC operation is used.

Parameters:

identifier (type: xs:string): Identifier for the schema list entry. Typically the module name or filename of the schema.

version (type: xs:string): Version of the schema supported.

format (type: SchemaFormat): The data modeling language of the file/module.

Example: Query a specific schema on device using <get-schema> operation

```xml
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-schema xmlns="urn:ietf:params:xml:ns:netconf:state">
    <identifier>foo</identifier>
    <version>v1</version>
    <format>XSD</format>
  </get-schema>
</rpc>
```

The NETCONF server returns the XML schema (XSD).

```xml
<rpc-reply message-id="101"
            xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data><![CDATA[
  <!-- Contents of foo schema would be returned here -->
  <xs:schema>
    <!-- Contents of foo schema would be returned here -->
  </xs:schema>]]>
</data>
</rpc-reply>
```

3.2. NETCONF Schema List Retrieval (<get> monitoring data)

A NETCONF client retrieves the list of supported XML Schema from a NETCONF server using the <get> RPC request. The list of available XML Schema is retrieved by requesting the <schema> subtree via a <get> operation. Available schema for the requesting session are
returned in the reply containing the <identifier>,<version> and <location> elements. Since the same schema may be supported in multiple locations and/or have multiple versions and/or multiple formats no particular attribute is unique. An empty reply is returned if there are no available schema.

The response data can be used to determine the available schema and their versions. The schema itself (ie. schema content) is not returned in the response. The details returned in the list SHOULD facilitate retrieval from a network location via a URL using a supported mechanism on the device such as ftp or http.

Additionally the ability to retrieve a schema via NETCONF SHOULD be supported. When a schema is available on the device and the schema-retrieval capability is supported by the NETCONF server a location value of ‘NETCONF’ MUST be used to indicate that it can be retrieved via NETCONF using the <get-schema> RPC described in section 3.1.
4. Examples

4.1. Retrieving schema list via <get>

```xml
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get>
    <filter type="subtree">
      <netconf xmlns="urn:ietf:params:xml:ns:netconf:state">
        <schemas/>
      </netconf>
    </filter>
  </get>
</ rpc>
```

The NETCONF server returns a list of data models available for retrieval.
<rpc-reply message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <netconf xmlns="urn:ietf:params:xml:ns:netconf:state">
      <schemas>
        <schema>
          <identifier>foo</identifier>
          <version>1.0</version>
          <format>XSD</format>
          <namespace>http://example.com/foo</namespace>
          <location>ftp://ftp.example.com/schemas/foo_1.0.xsd</location>
          <location>http://www.example.com/schema/foo_1.0.xsd</location>
          <location>NETCONF</location>
        </schema>
        <schema>
          <identifier>foo</identifier>
          <version>1.1</version>
          <format>XSD</format>
          <namespace>http://example.com/foo</namespace>
          <location>ftp://ftp.example.com/schemas/foo_1.1.xsd</location>
          <location>http://www.example.com/schema/foo_1.1.xsd</location>
          <location>NETCONF</location>
        </schema>
        <schema>
          <identifier>bar</identifier>
          <version>2008-06-01</version>
          <format>YANG</format>
          <namespace>http://example.com/bar</namespace>
          <location>http://example.com/schema/bar-2008-06-01.yang</location>
          <location>NETCONF</location>
        </schema>
        <schema>
          <identifier>bar-types</identifier>
          <version>2008-06-01</version>
          <format>YANG</format>
          <namespace>http://example.com/bar</namespace>
          <location>http://example.com/schema/bar-types-2008-06-01.yang</location>
          <location>NETCONF</location>
        </schema>
      </schemas>
    </netconf>
  </data>
</rpc-reply>
4.2. Retrieving schema instances

Given the reply in the previous section, the following examples illustrate the availability of 'foo', 'bar', and 'bar-types' schema at multiple locations, with multiple formats, and in multiple locations.

1. foo, version 1.0 in XSD format:
   a. Via FTP using location
      ftp://ftp.example.com/schemas/foo_1.0.xsd
   b. Via HTTP using location
      http://www.example.com/schema/foo/1.0.xsd
   c. Via <get-schema> using identifier, version, and format parameters.

   <rpc message-id="102"
       xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
     <get-schema xmlns="urn:ietf:params:xml:ns:netconf:state">
       <identifier>foo</identifier>
       <version>1.0</version>
       <format>XSD</format>
     </get-schema>
   </rpc>

2. bar, version 2008-06-01 in YANG format:
   a. Via HTTP using location
      http://example.com/schema/bar-2008-06-01.yang
   b. Via <get-schema> using identifier, version, and format parameters:

   <rpc message-id="103"
       xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
     <get-schema xmlns="urn:ietf:params:xml:ns:netconf:state">
       <identifier>bar</identifier>
       <version>2008-06-01</version>
       <format>YANG</format>
     </get-schema>
   </rpc>

3. bar-types, version 2008-06-01 in YANG format:
a. Via <get-schema> using identifier, version, and format parameters:

```xml
<rpc message-id="104"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-schema xmlns="urn:ietf:params:xml:ns:netconf:state">
    <identifier>bar-types</identifier>
    <version>2008-06-01</version>
    <format>YANG</format>
  </get-schema>
</rpc-reply>
```
5. XSD Schema

5.1. NETCONF Monitoring Schema

The data model described in this memo is defined in the following XSD.

```xml
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0"
           xmlns:ncEvent="urn:ietf:params:xml:ns:netconf:notification:1.0"
           xmlns:inet="urn:ietf:params:xml:ns:inet-types"
           targetNamespace="urn:ietf:params:xml:ns:netconf:state"
           elementFormDefault="qualified">

  <!-- import base netconf definitions -->
  <xs:import namespace="urn:ietf:params:xml:ns:netconf:base:1.0"
               schemaLocation="netconf.xsd"/>

  <xs:import namespace="urn:ietf:params:xml:ns:netconf:notification:1.0"
               schemaLocation="notification.xsd"/>

  <xs:import namespace="urn:ietf:params:xml:ns:inet-types"
               schemaLocation="inet-ip.xsd"/>

  <!-- Model -->
  <xs:element name="netconf">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="capabilities" minOccurs="1" maxOccurs="1">
          <xs:annotation>
            <xs:documentation>
              The list of currently provided NETCONF capabilities.
              This may be different than those exchanged during
              session setup (i.e. hello).
            </xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element name="configurations" minOccurs="1" maxOccurs="1">
          <xs:annotation>
            <xs:documentation>
              The list of currently configured NETCONF configurations.
              This may be different than those exchanged during
              session setup (i.e. hello).
            </xs:documentation>
          </xs:annotation>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```
Contains the NETCONF configurations data stores available on the device.

Contains the list of supported schemas on the device. I.e. NETCONF data models, interface descriptions, etc.

Contains the details on active sessions on the device. Covers both NETCONF and non-NETCONF management sessions.

Contains the details on active event stream
subscriptions. Includes active replay streams, if supported.
</xs:documentation>
</xs:annotation>
<xs:complexType>
 <xs:sequence>
  <xs:element name="subscription" type="NetconfSubscription"
   minOccurs="0" maxOccurs="unbounded"/>
 </xs:sequence>
</xs:complexType>

<xs:element name="statistics" type="ManagementStatistics"
  minOccurs="0" maxOccurs="1">
 <xs:annotation>
  <xs:documentation>
   Contains management metrics for the NETCONF server.
  </xs:documentation>
 </xs:annotation>
</xs:element>
</xs:sequence>
</xs:complexType>

<!-- Complex type definitions -->
<xs:complexType name="ConfigurationDatastore">
 <xs:annotation>
  <xs:documentation>
   Contains information on the datastores available on the NETCONF server including locks.
  </xs:documentation>
 </xs:annotation>
 <xs:sequence>
  <xs:element name="name" type="NETCONFDatastoreType">
   <xs:annotation>
    <xs:documentation>
     enumeration of supported datastores; candidate, running, startup
    </xs:documentation>
   </xs:annotation>
  </xs:element>
  <xs:element name="locks" minOccurs="0">
   <xs:annotation>
    <xs:documentation>
     An indication of whether a resource is locked or unlocked. If locked, additional information about
     the locking such as user an time stamp is provided.
    </xs:documentation>
   </xs:annotation>
  </xs:element>
 </xs:sequence>
</xs:complexType>
<xs:complexType>
  <xs:choice minOccurs="0">
    <xs:element name="globalLock" type="GlobalLock">
      <xs:annotation>
        <xs:documentation>
          Present if a global lock is set.
        </xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="partialLocks" type="PartialLock" maxOccurs="unbounded">
      <xs:annotation>
        <xs:documentation>
          Present if at least one partial lock is set.
        </xs:documentation>
      </xs:annotation>
    </xs:element>
  </xs:choice>
</xs:complexType>

<xs:complexType name="SchemaEntry">
  <xs:annotation>
    <xs:documentation>
      Contains the information used to describe the schema.
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="identifier" type="xs:string">
      <xs:annotation>
        <xs:documentation>
          Identifier to uniquely reference the schema.
        </xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="version" type="xs:string">
      <xs:annotation>
        <xs:documentation>
          Version of the schema supported. Multiple versions can be supported simultaneously.
        </xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="format" type="SchemaFormat">
      <xs:annotation>
<xs:documentation>
    Schema language for the file/module. Eg: XSD, YANG, RNG.
</xs:documentation>
</xs:annotation>

<xs:element name="namespace" type="xs:anyURI">
    <xs:annotation>
        <xs:documentation>
            The XML namespace defined by the data model.
        </xs:documentation>
    </xs:annotation>
</xs:element>

<xs:element name="location" minOccurs="0" maxOccurs="unbounded">
    <xs:annotation>
        <xs:documentation>
            A location from which the schema can be retrieved. Can be either on the network device retrievable explicitly via the get-schema netconf operation (denoted by the value ‘NETCONF’) or some network location (i.e. URL).
        </xs:documentation>
    </xs:annotation>
</xs:element>

<xs:simpleType>
    <xs:union>
        <xs:simpleType>
            <xs:restriction base="xs:string">
                <xs:enumeration value="NETCONF"/>
            </xs:restriction>
        </xs:simpleType>
        <xs:simpleType>
            <xs:restriction base="xs:anyURI"/>
        </xs:simpleType>
    </xs:union>
</xs:simpleType>

<xs:complexType name="ManagementSession">
    <xs:annotation>
        <xs:documentation>
            Contains information related to management sessions on the device.
        </xs:documentation>
    </xs:annotation>
    <xs:sequence>
        <xs:element name="sessionId" type="netconf:SessionId">
            <xs:annotation>
                
            </xs:annotation>
        </xs:element>
    </xs:sequence>
</xs:complexType>
<xs:documentation>
Unique identifier for the session instance.
</xs:documentation>
</xs:element>
<xs:element name="transport" type="TransportType">
<xs:annotation>
<xs:documentation>
Identifies the transport for each session, e.g.
"Console", "SSH".
</xs:documentation>
</xs:annotation>
</xs:element>

<xs:element name="protocol" type="ProtocolType">
<xs:annotation>
<xs:documentation>
Identifies the protocol being used for each session,
e.g. "NETCONF", "CLI", "WebUI".
</xs:documentation>
</xs:annotation>
</xs:element>

<xs:element name="username" type="xs:string">
<xs:annotation>
<xs:documentation>
Session owner.
</xs:documentation>
</xs:annotation>
</xs:element>

<xs:element name="sourceHost" type="inet:host">
<xs:annotation>
<xs:documentation>
Client for the session.
</xs:documentation>
</xs:annotation>
</xs:element>

<xs:element name="loginTime" type="xs:dateTime">
<xs:annotation>
<xs:documentation>
Time at which the session was established.
</xs:documentation>
</xs:annotation>
</xs:element>
</xs:sequence>
</xs:complexType>
<xs:complexType name="NetconfSubscription">
   <xs:documentation>
      Contains information on the active subscriptions on the NETCONF server. Subscriptions which have ended are not reported.
   </xs:documentation>
   <xs:sequence>
      <xs:element name="sessionId" type="netconf:SessionId"/>
      <xs:element name="stream" type="ncEvent:streamNameType"/>
      <xs:element name="filter" type="netconf:filterInlineType"/>
      <xs:element name="startTime" type="xs:dateTime" minOccurs="0"/>
      <!-- stopTime is optional on servers with replay support -->
      <xs:element name="stopTime" type="xs:dateTime" minOccurs="0"/>
      <xs:element name="messagesSent" type="xs:unsignedInt"/>
   </xs:sequence>
</xs:complexType>

<xs:complexType name="ManagementStatistics">
   <xs:documentation>
      Contains management statistics for the NETCONF server.
   </xs:documentation>
   <xs:sequence>
      <xs:element name="netconfStartTime" type="xs:dateTime">
      <xs:annotation>
         <xs:documentation>
            Date and time at which the NETCONF server process was started. Allows for calculation of time interval for reported metrics.
         </xs:documentation>
      </xs:annotation>
      </xs:element>
      <xs:element name="inSessions" type="xs:unsignedInt">
      <xs:annotation>
         <xs:documentation>
            The total number of NETCONF sessions started towards the NETCONF peer.
            inSessions - inBadHellos = ‘number of correctly started netconf sessions’
         </xs:documentation>
      </xs:annotation>
      </xs:element>
      <xs:element name="inXMLParseErrors" type="xs:unsignedInt">
      <xs:annotation>
         <xs:documentation>
         </xs:documentation>
      </xs:annotation>
      </xs:element>
   </xs:sequence>
</xs:complexType>
The total number of messages that were unparsable and thus ignored. This covers both unparsable 'hello' and 'rpc' messages.

The total number of sessions silently dropped because an invalid 'hello' message was received. This includes hello messages with a 'session-id' attribute, bad namespace, and bad capability declarations.

The total number of rpc requests received.

The total number of rpcs which were parsed correctly, but couldn't be serviced because they contained non-conformant XML, e.g. missing a mandatory parameter.

The total number of rpcs which were parsed correctly, but couldn't be serviced because they were not supported by the agent.

The total number of 'rpc-reply' messages sent.
<xs:element name="outRpcErrors" type="xs:unsignedInt">
    <xs:annotation>
        <xs:documentation>
            The total number of 'rpc-reply' messages with 'rpc-error' sent.
        </xs:documentation>
    </xs:annotation>
</xs:element>

<xs:element name="outNotifications" type="xs:unsignedInt">
    <xs:annotation>
        <xs:documentation>
            The total number of 'notification' messages sent.
        </xs:documentation>
    </xs:annotation>
</xs:element>
</xs:sequence>
</xs:complexType>

<xs:complexType name="NETCONFDatastoreType">
    <xs:annotation>
        <xs:documentation>
            Enumeration of possible NETCONF datastore types.
        </xs:documentation>
    </xs:annotation>
    <xs:choice>
        <xs:element name="startup" type="xs:string"/>
        <xs:element name="candidate" type="xs:string"/>
        <xs:element name="running" type="xs:string"/>
    </xs:choice>
</xs:complexType>

<xs:complexType name="GlobalLock">
    <xs:sequence>
        <xs:element name="lockedBySession" type="netconf:SessionId">
            <xs:annotation>
                <xs:documentation>
                    The session Id which holds the lock.
                </xs:documentation>
            </xs:annotation>
        </xs:element>
        <xs:element name="lockedTime" type="xs:dateTime">
            <xs:annotation>
                <xs:documentation>
                    Date and Time the lock was acquired.
                </xs:documentation>
            </xs:annotation>
        </xs:element>
    </xs:sequence>
</xs:complexType>
    </xs:complexType>

    <xs:complexType name="PartialLock">
    <xs:sequence>
        <xs:element name="lockId" type="xs:unsignedInt">
            <xs:annotation>
                <xs:documentation>
                    For a partial lock this provides the lock id.
                </xs:documentation>
            </xs:annotation>
        </xs:element>
        <xs:element name="lockedBySession" type="netconf:SessionId">
            <xs:annotation>
                <xs:documentation>
                    The session Id which holds the lock.
                </xs:documentation>
            </xs:annotation>
        </xs:element>
        <xs:element name="lockedTime" type="xs:dateTime">
            <xs:annotation>
                <xs:documentation>
                    Date and Time the lock was acquired.
                </xs:documentation>
            </xs:annotation>
        </xs:element>
        <xs:element name="select" maxOccurs="unbounded">
            <xs:annotation>
                <xs:documentation>
                    The xpath expression which was used to request the lock.
                </xs:documentation>
            </xs:annotation>
        </xs:element>
        <xs:element name="lockedNodes" minOccurs="0" maxOccurs="unbounded">
            <xs:annotation>
                <xs:documentation>
                    The list of instance-identifiers (i.e. the locked nodes).
                </xs:documentation>
            </xs:annotation>
        </xs:element>
    </xs:sequence>
    </xs:complexType>
<!-- Simple types -->
<xs:simpleType name="SchemaFormat">
  <xs:annotation>
    <xs:documentation>
      Format of the schema.
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:enumeration value="XSD"/>
    <xs:enumeration value="YANG"/>
    <xs:enumeration value="RNG"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="TransportType">
  <xs:annotation>
    <xs:documentation>
      Transport choices.
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:enumeration value="SSH"/>
    <xs:enumeration value="SSL"/>
    <xs:enumeration value="Console"/>
    <xs:enumeration value="HTTP"/>
    <xs:enumeration value="HTTPS"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="ProtocolType">
  <xs:annotation>
    <xs:documentation>
      Protocol choices.
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:enumeration value="CLI"/>
    <xs:enumeration value="NETCONF"/>
    <xs:enumeration value="WebUI"/>
  </xs:restriction>
</xs:simpleType>

<!-- RPC Definitions -->
<xs:element name="get-schema" substitutionGroup="netconf:rpcOperation">
  <xs:annotation>
    <xs:documentation>
      RPC definition: &lt;get-schema&gt;
    </xs:documentation>
  </xs:annotation>
</xs:element>
5.2. inet:host schema

The complex type ManagementSession contains element sourceHost of type inet:host. The following schema defines the referenced type.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="urn:ietf:params:xml:ns:inet-types"
  xmlns:inet="urn:ietf:params:xml:ns:inet-types"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified">

  <xs:simpleType name="ipAddress">
    <xs:documentation>
      The ipAddress type represents an IP address and is IP version neutral. The format of the textual representations implies the IP version.
    </xs:documentation>
  </xs:simpleType>

  <xs:union>
    <xs:simpleType>
      <xs:restriction base="inet:ipv4Address"/>
    </xs:simpleType>
    <xs:simpleType>
      <xs:restriction base="inet:ipv6Address"/>
    </xs:simpleType>
  </xs:union>
</xs:schema>
```
</xs:union>
</xs:simpleType>

<xs:simpleType name="ipv4Address">
  <xs:annotation>
    <xs:documentation>
      The ipv4Address type represents an IPv4 address in
dotted-quad notation. The IPv4 address may include
      a zone index, separated by a % sign.
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:pattern value="((0-1)?[0-9-][0-9-]2[0-4][0-9-]25[0-5]\.)3
      (0-1)?[0-9-][0-9-]2[0-4][0-9-]25[0-5]
      (%[pN]\p{L}]+)?"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="ipv6Address">
  <xs:annotation>
    <xs:documentation>
      The ipv6Address type represents an IPv6 address in
      full, mixed, shortened and shortened mixed notation.
      The IPv6 address may include a zone index, separated
      by a % sign.
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:pattern value="((((0-9a-fA-F){1,4}:){7})((0-9a-fA-F){1,4})
      (%[pN]\p{L}]+)?
    |((((0-9a-fA-F){1,4}:){6})((0-9){1,3}.
      [0-9]{1,3}\.[0-9]{1,3}([0-9]{1,3})
      (%[pN]\p{L}]+)?
    |((((0-9a-fA-F){1,4})*([0-9a-fA-F]{1,4})*::)
      ([0-9a-fA-F]{1,4})*([0-9a-fA-F]{1,4})*
      (%[pN]\p{L}]+)?
    |((((0-9a-fA-F){1,4})*([0-9a-fA-F]{1,4})::*::)
      ([0-9a-fA-F]{1,4})*([0-9a-fA-F]{1,4})*
      ([0-9]{1,3}.
      [0-9]{1,3}\.[0-9]{1,3}([0-9]{1,3})
      (%[pN]\p{L}]+)?")/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="domainName"
The domainName type represents a DNS domain name. The name SHOULD be fully qualified whenever possible.

The description clause of objects using the domainName type MUST describe how (and when) these names are resolved to IP addresses.

Note that the resolution of a domainName value may require to query multiple DNS records (e.g., A for IPv4 and AAAA for IPv6). The order of the resolution process and which DNS record takes precedence depends on the configuration of the resolver.
6. Security Considerations

The NETCONF monitoring schema as defined in this document provides information about a NETCONF system that could be used to aid an attack on that system. Care should be taken to restrict access to this information as appropriate.
7. Acknowledgements

Thanks to the NETCONF working group for providing a much earlier draft of Schema to manage NETCONF and NETCONF Notifications that were cannibalized to produce this draft.
8. IANA Considerations

This document will require new URIs for the NETCONF XML namespace in the IETF XML registry [RFC3688]. Following the format in RFC 3688, the following registrations require consideration.

<table>
<thead>
<tr>
<th>Index</th>
<th>Capability Identifier</th>
</tr>
</thead>
</table>

Note, the proposed capability URNs are also compliant to section 10.3 of NETCONF [RFC4741].
9. Normative References


[RFC2119] Bradner, s., "Key words for RFCs to Indicate Requirements Levels", RFC 2119, March 1997.


Appendix A.  YANG module (non-normative)

The following YANG module is included as a reference only.  It is based on YANG definition at the time of publishing and is subject to change as a result of netmod work underway to refine YANG.  Further details on YANG and updated non-normative reference to this model are available at:
http://www.yang-central.org/twiki/bin/view/Main/YangExamples.

module netconf-state {
    namespace "urn:ietf:params:xml:ns:netconf:state";
    prefix "ns";

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

    organization
        "IETF";

    description
        "NETCONF Monitoring Module.  
        All elements in this module are read-only.";

    revision "2008-11-03" {
        description "Initial revision.";
    }

    typedef SessionId {
        type uint32 {
            range "1..max";
        }
        reference "rfc4741";
    }

    grouping NETCONFDatastoreType {
        description
            "Enumeration of possible NETCONF datastore types.";
        reference "rfc4741";
        choice datastore {
            mandatory true;
            leaf running {
                type empty;
            }
            leaf candidate {
                type empty;
            }
        }
    }
}
leaf startup {
    type empty;
}

typedef TransportType {
    type enumeration {
        enum "SSH";
        enum "SSL";
        enum "Console";
        enum "HTTP";
        enum "HTTPS";
    }
}

typedef ProtocolType {
    type enumeration {
        enum "CLI";
        enum "NETCONF";
        enum "WebUI";
    }
}

typedef SchemaFormat {
    type enumeration {
        enum "XSD" {
            reference "W3C REC REC-xmlschema-1-20041028";
        }
        enum "YANG" {
            reference "draft-ietf-netmod-yang";
        }
        enum "RNG" {
            reference "ISO/IEC 19757-2";
        }
    }
}

container netconf {
    config false;

    container capabilities {
        description
            "The list of currently provided NETCONF capabilities. This
            may be different than those exchanged during session setup
            (i.e. hello).";
        leaf-list capability {
            type inet:uri;
min-elements 1;
}
}

container configurations {
    description
        "List of NETCONF configuration datastores (e.g. running,
        startup, candidate) supported on this device and related
        information.";
    list configuration {
        container name {
            uses NETCONFDatastoreType;
        }
    }
    container locks {
        description
            "An indication of whether a resource is locked or
            unlocked. If locked, additional information about
            the locking such as user an time stamp is provided.";
    }
}

grouping LockInfo {
    leaf lockedBySession {
        type SessionId;
        description
            "The session ID of the session that has locked
            this resource.";
    }
    leaf lockedTime {
        type yang:date-and-time;
        description
            "The date and time of when the resource was
            locked.";
    }
}

choice lockType {
    container globalLock {
        description
            "Present if the global lock is set.";
        uses LockInfo;
    }
    list partialLocks {
        key lockId;
        description
            "Present if at least one partial lock is set.";
        leaf lockId {
            type uint32;
        }
    }
}
uses LockInfo;
leaf-list select {
    type string;
    min-elements 1;
    description
    "The xpath expression which was used to request
    the lock.";
}
leaf-list lockedNodes {
    type instance-identifier;
    description
    "The list of instance-identifiers (i.e. the
    locked nodes).";
}

container schemas {
    list schema {
        key "identifier format version";
        leaf identifier {
            type string;
            description
            "Identifier to uniquely reference the schema";
        }
        leaf version {
            type string;
            description
            "Version of the schema supported. Multiple versions can be
            supported simultaneously.";
        }
        leaf format {
            type SchemaFormat;
            description
            "Schema language for the file/module.";
        }
        leaf namespace {
            type inet:uri;
            description
            "The XML namespace defined by the data model.";
        }
        leaf location {
            type union {
                type enumeration {
                    enum "NETCONF";
                }
            }
        }
    }
}
type inet:uri;

description
"A location from which the schema can be retrieved. Can be either on the network device retrievable explicitly via the get-schema netconf operation (denoted by the value 'NETCONF') or some network location (i.e. URL).";

container sessions {
    description
    "List of NETCONF sessions currently active on this device."
    list session {
        key sessionId;
        leaf sessionId {
            type SessionId;
        }
        leaf transport {
            type TransportType;
        }
        leaf protocol {
            type ProtocolType;
        }
        leaf username {
            type string;
        }
        leaf sourceHost {
            type inet:host;
        }
        leaf loginTime {
            type yang:date-and-time;
            description
            "Time at which the session was established."
        }
    }
}

container subscriptions {
    description
    "Contains information on the active subscriptions on the NETCONF server. Subscriptions which have ended are not reported."
    list subscription {
        key sessionId;
        description
        ""
"Information about Netconf Notification Subscriptions."
leaf sessionId {
    type SessionId;
    description
        "The session id associated with this subscription."
}
leaf stream {
    type string;
    description
        "The stream associated with this subscription."
}
anyxml filter {
    description
        "The filters associated with this subscription."
}
leaf startTime {
    type yang:date-and-time;
    description
        "The startTime parameter from the create-subscription
         invokation, if it was present."
}
leaf stopTime {
    type yang:date-and-time;
    description
        "The stopTime parameter from the create-subscription
         invokation, if it was present."
}
leaf messagesSent {
    type yang:zero-based-counter32;
    description
        "A count of event notifications sent along
         this connection since the subscription was
         created."
}

ccontainer statistics {
    leaf netconfStartTime {
        type yang:date-and-time;
        description
            "Date and time at which the NETCONF server process was
             started. Allows for calculation of time interval for
             reported metrics."
    }
    leaf inSessions {
        type yang:zero-based-counter32;
        description
            ""
"The total number of NETCONF sessions started towards the NETCONF peer.

\[
in\text{Sessions} - in\text{BadHellos} = \text{number of correctly started netconf sessions}
\]

leaf inXMLParseErrors {
  type yang:zero-based-counter32;
  description
  "The total number of messages that were unparsable and thus ignored. This covers both unparsable 'hello' and 'rpc' messages."
}
leaf inBadHellos {
  type yang:zero-based-counter32;
  description
  "The total number of sessions silently dropped because an invalid 'hello' message was received. This includes hello messages with a 'session-id' attribute, bad namespace, and bad capability declarations."
}
leaf inRpcs {
  type yang:zero-based-counter32;
  description
  "The total number of rpc requests received."
}
leaf inBadRpcs {
  type yang:zero-based-counter32;
  description
  "The total number of rpcs which were parsed correctly, but couldn't be serviced because they contained non-conformant XML, e.g. missing a mandatory parameter."
}
leaf inNotSupportedRpcs {
  type yang:zero-based-counter32;
  description
  "The total number of rpcs which were parsed correctly, but couldn't be serviced because they were not supported by the agent."
}
leaf outRpcReplies {
  type yang:zero-based-counter32;
  description
  "The total number of 'rpc-reply' messages sent."
}
leaf outRpcErrors {
  type yang:zero-based-counter32;
  description
"The total number of 'rpc-reply' messages with 'rpc-error' sent."
}
leaf outNotifications {
    type yang:zero-based-counter32;
    description
      "The total number of 'notifications' messages sent.";
}
}
}
rpc get-schema {
  input {
    leaf identifier {
      type string;
      mandatory true;
    }
    leaf version {
      type string;
      mandatory true;
    }
    leaf format {
      type SchemaFormat;
      mandatory true;
    }
  }
  output {
    anyxml data {
      description "Contains the schema content.";
    }
  }
}
Authors’ Addresses

Mark Scott
Nortel
3500 Carling Ave
Nepean, Ontario  K2H 8E9
Canada

Email: markscot@nortel.com

Sharon Chisholm
Nortel
3500 Carling Ave
Nepean, Ontario  K2H 8E9
Canada

Email: schishol@nortel.com

Martin Bjorklund
Tail-f Systems
Klara Norra Kyrkogata 31
SE-111 22 Stockholm,
Sweden

Email: mbj@tail-f.com