

Network Working Group
Internet Draft

Intended status: Experimental
Expires: November 2015

T. Mizrahi
Y. Moses
Technion, Israel Institute of Technology
May 7, 2015

Time Capability in NETCONF
draft-mm-netconf-time-capability-05.txt

Abstract

This document defines a capability-based extension to the Network Configuration Protocol (NETCONF) that allows time-triggered configuration and management operations. This extension allows NETCONF clients to invoke configuration updates according to scheduled times, and allows NETCONF servers to attach timestamps to the data they send to NETCONF clients.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of [BCP 78](#) and [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/lid-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 November 7, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents

(<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Introduction	3
2.	Conventions used in this document	3
2.1.	Keywords	3
2.2.	Abbreviations	3
2.3.	Terminology	3
3.	Using Time in NETCONF	4
3.1.	The Time Capability in a Nutshell	4
3.2.	Notifications and Cancellation Messages	6
3.3.	Synchronization Aspects	8
3.4.	Scheduled Time Format	9
3.5.	Scheduling Tolerance	9
3.6.	Near Future Scheduling vs. Far Future Scheduling	10
3.7.	Time Interval Format	11
4.	Time Capability	11
4.1.	Overview	11
4.2.	Dependencies	12
4.3.	Capability Identifier	12
4.4.	New Operations	12
4.5.	Modifications to Existing Operations	12
4.6.	Interactions with Other Capabilities	13
5.	Examples	13
5.1.	<scheduled-time> Example	13
5.2.	<get-time> Example	14
5.3.	Error Example	15
6.	Security Considerations	16
6.1.	General Security Considerations	16
6.2.	YANG Module Security Considerations	16
7.	IANA Considerations	17
8.	Acknowledgments	17
9.	References	18
9.1.	Normative References	18
9.2.	Informative References	18
Appendix A.	YANG Module for the Time Capability	19

1. Introduction

The Network Configuration Protocol (NETCONF) defined in [[RFC6241](#)] provides mechanisms to install, manipulate, and delete the configuration of network devices. NETCONF allows clients to configure and monitor NETCONF servers using remote procedure calls (RPC).

NETCONF, as defined in [[RFC6241](#)], is asynchronous; when a client invokes an RPC, it has no control over the time at which the RPC is executed, nor does it have any feedback from the server about the execution time.

Time-based configuration ([[HotSDN](#)], [[TimeTR](#)]) can be a useful tool that enables an entire class of coordinated and scheduled configuration procedures. Time-triggered configuration allows coordinated network updates in multiple devices; a client can invoke a coordinated configuration change by sending RPCs to multiple servers with the same scheduled execution time. A client can also invoke a time-based sequence of updates by sending n RPCs with n different update times, T_1 , T_2 , ..., T_n , determining the order in which the RPCs are executed.

This memo defines the time capability in NETCONF. This extension allows clients to determine the scheduled execution time of RPCs they send. It also allows a server that receives an RPC to report its actual execution time to the client.

2. Conventions used in this document

2.1. Keywords

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 [[RFC2199](#)].

2.2. Abbreviations

NETCONF Network Configuration Protocol

[2.3](#). Terminology

- o Capability [[RFC6142](#)]: A functionality that supplements the base NETCONF specification.

- o Client [[RFC6142](#)]: Invokes protocol operations on a server. In addition, a client can subscribe to receive notifications from a server.
- o Execution time: The execution time of an RPC is defined as the time at which a server completes the execution of an RPC.
- o Scheduled time: The scheduled time of an RPC is the time at which the RPC should be invoked. The scheduled time is determined by the client, and enforced by the server.
- o Server [[RFC6142](#)]: Executes protocol operations invoked by a client. In addition, a server can send notifications to a client.

[3](#). Using Time in NETCONF

[3.1](#). The Time Capability in a Nutshell

The `:time` capability provides two main functions:

- o Scheduling:
When a client sends an RPC to a server, the RPC message MAY include a scheduled time, T_s (see Figure 1). The server then executes the RPC at the scheduled time T_s , and once completed the server can respond with an RPC reply message.
- o Reporting:
When a client sends an RPC to a server, the RPC message MAY include a `get-time` element (see Figure 2), requesting the server to return the execution time of the RPC. In this case, after the server performs the RPC it responds with an RPC reply that includes the execution time, T_e .

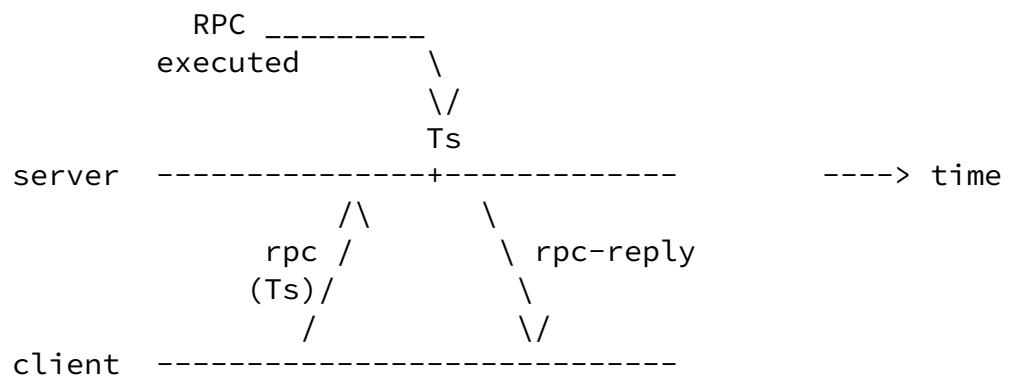


Figure 1 Scheduled RPC

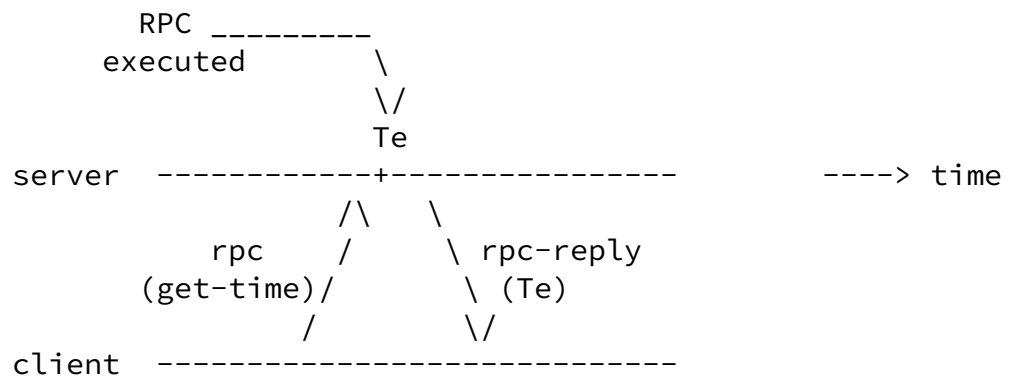


Figure 2 Reporting the Execution Time of an RPC

The two scenarios discussed above imply that a third scenario can

subscribed to the NETCONF notifications [[RFC6470](#)]; as illustrated in Figure 4, when the server receives a scheduled RPC it sends a notification to the client.

The <netconf-scheduled-message> notification includes a <schedule-id> element. The <schedule-id> is a unique identifier that the server assigns to every scheduled RPC it receives. Thus, a client can keep track of all the pending scheduled RPCs; a client can uniquely identify a scheduled RPC by the tuple {server, schedule-id}.

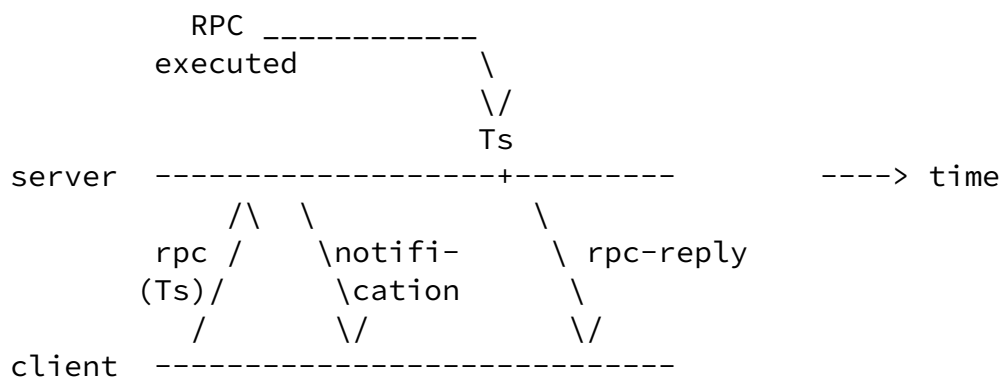


Figure 4 Scheduled RPC with Notification

Cancellation Messages

A client can cancel a scheduled RPC by sending a <cancel-schedule> RPC. The <cancel-schedule> RPC includes the <schedule-id> of the scheduled RPC that needs to be cancelled.

The <cancel-schedule> RPC, defined in this document, can be used to perform a coordinated all-or-none procedure, where either all the servers perform the operation on schedule, or the operation is aborted.

Example. The client sends scheduled RPC messages to server 1 and server 2, both scheduled to Ts. Server 1 sends a notification that indicates it has successfully scheduled the RPC, while server 2 replies with an unknown-element error [[RFC6241](#)] that indicates that it does not support the time capability. The client sends a <cancel-schedule> RPC to server 1, and receives an rpc-reply. The message exchange between the client and server 1 in this example is illustrated in Figure 5.

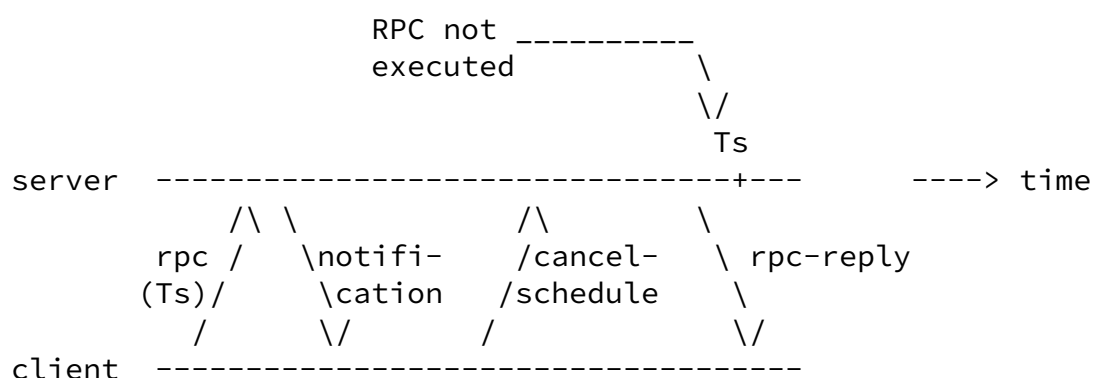


Figure 5 Cancellation Message

[3.3.](#) Synchronization Aspects

The time capability defined in this document requires clients and servers to maintain clocks. It is assumed that clocks are synchronized by a method that is outside the scope of this document, e.g., [[NTP](#)] or [[IEEE1588](#)].

This document does not define any requirements pertaining to the

degree of accuracy of performing scheduled RPCs. Note that two factors affect how accurately the server can perform a scheduled RPC; one factor is the accuracy of the clock synchronization method used to synchronize the clients and servers, and the second factor is the server's ability to execute real-time configuration changes, which greatly depends on how it is implemented. Typical networking devices are implemented by a combination of hardware and software. While the execution time of a hardware module can typically be predicted with a high level of accuracy, the execution time of a software module may be variable and hard to predict. A configuration update would typically require the server's software to be involved, thus affecting how accurately the RPC can be scheduled.

Another important aspect of synchronization, is monitoring; a client should be able to check whether a server is synchronized to a reference time source. Typical synchronization protocols, such as the Network Time Protocol ([\[NTP\]](#), [\[RFC5907\]](#)) provide the means to verify that a clock is synchronized to a time reference by querying its Management Information Base (MIB). The get-time feature defined in this document (see Figure 2) allows a client to obtain a rough estimate of the time offset between the client's clock and the server's clock.

Since servers do not perform configuration changes instantaneously, the processing time of an RPC should not be overlooked. The scheduled time always refers to the start time of the RPC, and the execution time always refers to its completion time.

[3.4. Scheduled Time Format](#)

The scheduled time and execution time fields in RPC messages use a common time format field.

The time format used in this document is the date-and-time format, that is defined in [Section 5.6 of \[RFC3339\]](#) and in [Section 3 of \[RFC6021\]](#).

```
leaf scheduled-time {  
  type yang:date-and-time;  
  description
```

```

    "The time at which the RPC is scheduled to be performed.";
}

leaf execution-time {
    type yang:date-and-time;
    description
        "The time at which the RPC was executed.";
}

```

3.5. Scheduling Tolerance

When a client sends an RPC that is scheduled to T_s , the server MUST verify that the value T_s is not too far in the past or in the future. As illustrated in Figure 6, the server verifies that T_s is within the scheduling tolerance range.

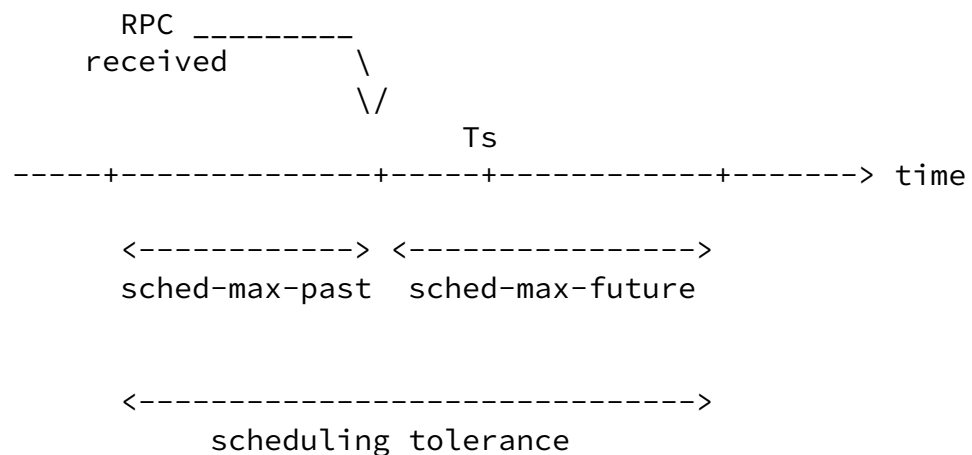


Figure 6 Scheduling Tolerance

The scheduling tolerance is determined by two parameters, sched-max-future and sched-max-past. These two parameters use the time-interval format ([Section 3.7.](#)), and their default value is 15 seconds.

If the scheduled time, T_s is within the scheduling tolerance range, the scheduled RPC is performed; if T_s occurs in the past and within the scheduling tolerance, the server performs the RPC as soon as possible, whereas if T_s is a future time, the server performs the RPC at T_s .

If T_s is not within the scheduling tolerance range, the scheduled RPC is discarded, and the server responds with an error message [RPC6241] with a bad-element error-tag. An example is provided in [Section 5.3.](#)

[3.6.](#) Near Future Scheduling vs. Far Future Scheduling

The scheduling bound defined by sched-max-future guarantees that every scheduled RPCs is restricted to a near future scheduling time.

The scheduling mechanism defined in this document is intended for near future scheduling, on the order of seconds. Far future scheduling is outside the scope of this document.

The challenge in far future scheduling is that during the long period between the time at which the RPC is sent and the time at which it is scheduled to be executed various external events may occur, e.g., the client may fail or reboot, or the client access permissions may be changed. In these cases if the server performs the scheduled operation it may perform an action that is inconsistent with the

current network policy, or inconsistent with the currently active clients.

Near future scheduling guarantees that external events such as the examples above have a low probability of occurring during the sched-max-future period, and even when they do, the period of inconsistency is limited to sched-max-future, which is a short period of time.

[3.7.](#) Time Interval Format

The time-interval format is used for representing the length of a time interval, and is based on the date-and-time format. It is used

for representing the scheduling tolerance parameters, as described in the previous section.

While the date-and-time type uniquely represents a specific point in time, the time-interval type defined below can be used to represent the length of a time interval without specifying a specific date.

The time-interval type is defined as follows:

```
typedef time-interval {  
    type string {  
        pattern '\d{2}:\d{2}:\d{2}(\.\d+)?';  
    }  
}
```

[4. Time Capability](#)

The structure of this section is as defined in [Appendix D of \[RFC6241\]](#).

[4.1. Overview](#)

A server that supports the time capability can perform time-triggered operations as defined in this document.

A server implementing the :time capability:

- o MUST support the ability to receive <rpc> messages that include a time element, and perform a time-triggered operation accordingly.
- o MUST support the ability to include a time element in the <rpc-reply> messages that it transmits.

[4.2. Dependencies](#)

With-defaults Capability

The time capability YANG module (Appendix A.) uses default values, and thus it is assumed that the with-defaults capability [\[RFC6243\]](#) is supported.

[4.3.](#) Capability Identifier

The `:time` capability is identified by the following capability string (to be assigned by IANA – see [Section 0](#)):

`urn:ietf:params:netconf:capability:time:1.0`

[4.4.](#) New Operations

`<cancel-schedule>`

The `cancel-schedule` RPC is used for cancelling an RPC that was previously scheduled.

A `cancel-schedule` RPC MUST include the `<cancelled-message-id>` element, which specifies the message ID of the scheduled RPC that needs to be cancelled.

A `cancel-schedule` RPC MAY include the `<get-time>` element. In this case the `rpc-reply` includes the `<execution-time>` element, specifying the time at which the scheduled RPC was cancelled.

[4.5.](#) Modifications to Existing Operations

Three new elements are added to all existing operations:

- o `<scheduled-time>`

This element is added to the input of each operation, indicating the time at which the server is scheduled to invoke the operation. Every `<rpc>` message MAY include the `<scheduled-time>` element. A server that supports the `:time` capability and receives an `<rpc>` message with a `<scheduled-time>` element MUST perform the operation as close as possible to the scheduled time.

The `<scheduled-time>` element uses the date-and-time format ([Section 3.4.](#)).

- o `<get-time>`

This element is added to the input of each operation. An `<rpc>` message MAY include a `<get-time>` element, indicating that the

server MUST include an <execution-time> in its corresponding <rpc-reply>.

o <execution-time>

This element is added to the output of each operation, indicating the time at which the server completed the operation. An <rpc-reply> MAY include the <execution-time> element. A server that supports the :time capability and receives an operation with the <get-time> element MUST include the execution time in its response.

The execution-time element uses the date-and-time format ([Section 3.4.](#)).

[4.6.](#) Interactions with Other Capabilities

Confirmed Commit Capability

The confirmed commit capability is defined in [Section 8.4 of \[RFC6241\]](#). According to [\[RFC6241\]](#), a confirmed <commit> operation MUST be reverted if a confirming commit is not issued within the timeout period (which by default is 600 seconds).

When the time capability is supported, and a confirmed <commit> operation is used with the <scheduled-time> element, the confirmation timeout MUST be counted from the scheduled time, i.e., the client begins the timeout measurement starting at the scheduled time.

[5.](#) Examples

[5.1.](#) <scheduled-time> Example

The following example extends the example presented in [Section 7.2 of \[RFC6241\]](#) by adding the time capability. In this example, the <scheduled-time> element is used to specify the scheduled execution time of the configuration update (as shown in Figure 1).

```
<rpc message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
```

```

    <scheduled-time
      xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-time">
      2015-10-21T04:29:00.235Z
    </scheduled-time>
  </config>
  <top xmlns="http://example.com/schema/1.2/config">
    <interface>
      <name>Ethernet0/0</name>
      <mtu>1500</mtu>
    </interface>
  </top>
</config>
</edit-config>
</rpc>

<rpc-reply message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ok/>
</rpc-reply>

```

5.2. <get-time> Example

The following example is similar to the one presented in [Section 5.1](#), except that in this example the client includes a <get-time> element in its RPC, and the server consequently responds with an <execution-time> element (as shown in Figure 2).

```

<rpc message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <get-time
      xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-time">
    </get-time>
  </edit-config>
  <top xmlns="http://example.com/schema/1.2/config">
    <interface>
      <name>Ethernet0/0</name>
      <mtu>1500</mtu>
    </interface>
  </top>
</rpc>

```

```

        </interface>
    </top>
</config>
</edit-config>
</rpc>

<rpc-reply message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ok/>
  <execution-time>
    2015-10-21T04:29:00.235Z
  </execution-time>
</rpc-reply>

```

[5.3.](#) Error Example

The following example presents a scenario in which the scheduled-time is not within the scheduling tolerance, i.e., it is too far in the past, and therefore an rpc-error is returned.

```

<rpc message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <scheduled-time
      xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-time">
      2010-10-21T04:29:00.235Z
    </scheduled-time>
    <config>
      <top xmlns="http://example.com/schema/1.2/config">
        <interface>
          <name>Ethernet0/0</name>
          <mtu>1500</mtu>
        </interface>
      </top>
    </config>
  </edit-config>
</rpc>

<rpc-reply message-id="101"

```

```
    xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <rpc-error>
    <error-type>application</error-type>
    <error-tag>bad-element</error-tag>
    <error-severity>error</error-severity>
    <error-info>
      <bad-element>scheduled-time</bad-element>
    </error-info>
  </rpc-error>
</rpc-reply>
```

[6. Security Considerations](#)

[6.1. General Security Considerations](#)

The security considerations of the NETCONF protocol in general are discussed in [\[RFC6241\]](#).

The usage of the time capability defined in this document can assist an attacker in gathering information about the system, such as the exact time of future configuration changes. Moreover, the time elements can potentially allow an attacker to learn information about the system's performance. Furthermore, an attacker that sends malicious RPC messages can use the time capability to amplify her attack; for example, by sending multiple RPC messages with the same scheduled time. It is important to note that the security measures described in [\[RFC6241\]](#) can prevent these vulnerabilities.

The time capability relies on an underlying time synchronization protocol. Thus, an attack against the time protocol can potentially compromise NETCONF when using the time capability. A detailed discussion about the threats against time protocols and how to mitigate them is presented in [\[TimeSec\]](#).

[6.2. YANG Module Security Considerations](#)

This memo defines a new YANG module, as specified in [Appendix A](#).

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 [\[RFC6242\]](#). The NETCONF access control model [\[RFC6536\]](#) provides the means to restrict access for particular NETCONF users to a pre-configured subset of all available NETCONF protocol operations and content.

This YANG module defines `<sched-max-future>` and `<sched-max-past>`, which are writable/creatable/deletable. These data nodes may be considered sensitive or vulnerable in some network environments. For instance, an attacker may attempt to maliciously configure these parameters to a small value, thereby causing all scheduled RPCs to be discarded.

This YANG module defines the `<cancel-schedule>` RPC. This RPC may be considered sensitive or vulnerable in some network environments; it may be used maliciously to attack servers by canceling their pending RPCs. It is thus important to control access to this operations; the authorisation level required to cancel an RPC should be the same as the level required to schedule it.

[7.](#) IANA Considerations

This document proposes to register the following capability identifier URN in the 'Network Configuration Protocol (NETCONF) Capability URNs' registry:

```
urn:ietf:params:netconf:capability:time:1.0
```

This document proposes to register the following XML namespace URN in the 'IETF XML registry', following the format defined in [\[RFC3688\]](#):

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

This document proposes to register a module name in the 'YANG Module Names' registry, defined in [\[RFC6020\]](#).

```
name: ietf-netconf-time
```

```
prefix: nct
```

```
namespace: urn:ietf:params:xml:ns:yang:ietf-netconf-time
```

```
RFC: TBD
```

[8.](#) Acknowledgments

The authors gratefully acknowledge Joe Marcus Clarke, Andy Bierman, Balazs Lengyel, Jonathan Hansford, John Heasley, Alon Schneider and Eylon Egozi for their insightful comments.

This work was supported in part by Israel Science Foundation grant ISF 1520/11.

This document was prepared using 2-Word-v2.0.template.dot.

[9.](#) References

[9.1.](#) Normative References

- [RFC2199] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.
- [RFC3339] Klyne, G., Ed. and C. Newman, "Date and Time on the Internet: Timestamps", [RFC 3339](#), July 2002.
- [RFC3688] Mealling, M., "The IETF XML Registry", [BCP 81](#), [RFC 3688](#), January 2004.
- [RFC6021] Schoenwaelder, J., "Common YANG Data Types", [RFC 6021](#), October 2010.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., Bierman, A., Ed., "Network Configuration Protocol (NETCONF)", [RFC 6241](#), June 2011.
- [RFC6470] Bierman, A., "Network Configuration Protocol (NETCONF) Base Notifications", [RFC 6470](#), February 2012.

[9.2.](#) Informative References

- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", [RFC 6242](#), June 2011.
- [RFC6243] Bierman, A., Lengyel, B., "With-defaults Capability for NETCONF", [RFC 6243](#), June 2011.
- [RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", [RFC 6536](#), March 2012.
- [HotSDN] Mizrahi, T., Moses, Y., "Time-based Updates in Software Defined Networks", the second workshop on hot topics in software defined networks (HotSDN), 2013.
- [IEEE1588] IEEE TC 9 Instrumentation and Measurement Society,

"1588 IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems Version 2", IEEE Standard, 2008.

Mizrahi, Moses

Expires November 7, 2015

[Page 18]

Internet-Draft

Time Capability in NETCONF

May 2015

- [NTP] Mills, D., Martin, J., Burbank, J., Kasch, W., "Network Time Protocol Version 4: Protocol and Algorithms Specification", [RFC 5905](#), June 2010.
- [RFC5907] Gerstung, H., Elliott, C., Haberman, B., "Definitions of Managed Objects for Network Time Protocol Version 4 (NTPv4)", [RFC 5907](#), June 2010.
- [TimeSec] Mizrahi, T., "Security Requirements of Time Protocols in Packet Switched Networks", [RFC 7384](#), October 2014.
- [TimeTR] Mizrahi, T., Moses, Y., "Time-based Updates in OpenFlow: A Proposed Extension to the OpenFlow Protocol", Technion - Israel Institute of Technology, technical report, CCIT Report #835, EE Pub No. 1792, 2013.
<http://tx.technion.ac.il/~dew/OFTimeTR.pdf>

[Appendix A.](#)

YANG Module for the Time Capability

This section is normative.

```
<CODE BEGINS> file "ietf-netconf-time@2015-05-07.yang"
```

```
module ietf-netconf-time {  
  
    namespace "urn:ietf:params:xml:ns:yang:ietf-netconf-time";  
  
    prefix nct;  
  
    import ietf-netconf { prefix nc; }  
  
    import ietf-yang-types { prefix yang; }  
  
    import ietf-netconf-monitoring { prefix ncm; }
```

```
organization
  "IETF";

contact
  "Editor: Tal Mizrahi
    <dew@tx.technion.ac.il>
  Editor: Yoram Moses
    <moses@ee.technion.ac.il>";
```

Mizrahi, Moses

Expires November 7, 2015

[Page 19]

Internet-Draft

Time Capability in NETCONF

May 2015

```
description
  "This module defines a capability-based extension to the
  Network Configuration Protocol (NETCONF) that allows
  time-triggered configuration and management operations.
  This extension allows NETCONF clients to invoke configuration
  updates according to scheduled times, and allows NETCONF
  servers to attach timestamps to the data they send to NETCONF
  clients.
```

Copyright (c) 2015 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>).";

```
revision 2015-05-07 {
  description
    "Initial version.";
  reference
    "draft-mm-netconf-time-capability:
    Time Capability in NETCONF";
}
```

```
typedef time-interval {
  type string {
    pattern '\d{2}:\d{2}:\d{2}(\.\d+)?';
```

```

    }
    description
        "Defines a time interval, up to 24 hours.";
}

```

```

grouping scheduling-tolerance-parameters {
    leaf sched-max-future {
        type time-interval;
        default 00:00:15.0;
        description
            "When the scheduled time is in the future, i.e., greater
            than the present time, this leaf defines the maximal

```

```

        difference between the scheduled time
        and the present time that the server is willing to
        accept. If the difference exceeds this number, the
        server responds with an error.";
    }

    leaf sched-max-past {
        type time-interval;
        default 00:00:15.0;
        description
            "When the scheduled time is in the past, i.e., less
            than the present time, this leaf defines the maximal
            difference between the present time
            and the scheduled time that the server is willing to
            accept. If the difference exceeds this number, the
            server responds with an error.";
    }

    description
        "Contains the parameters of the scheduling tolerance.";
}

```

```

// extending the get-config operation
augment /nc:get-config/nc:input {
    leaf scheduled-time {
        type yang:date-and-time;
        description
            "The time at which the RPC is scheduled to be performed.";
    }
}

```

```

    }

    leaf get-time {
        type empty;
        description
            "Indicates that the rpc-reply should include the
            execution-time.";
    }

    description
        "Adds the time element to <get-config>.";
}

```

```

augment /nc:get-config/nc:output {
    leaf execution-time {
        type yang:date-and-time;
        description
            "The time at which the RPC was executed.";
    }

    description
        "Adds the time element to <get-config>.";
}

augment /nc:get/nc:input {
    leaf scheduled-time {
        type yang:date-and-time;
        description
            "The time at which the RPC is scheduled to be performed.";
    }

    leaf get-time {
        type empty;
        description
            "Indicates that the rpc-reply should include the
            execution-time.";
    }

    description
        "Adds the time element to <get>.";
}

```

```

augment /nc:get/nc:output {
  leaf execution-time {
    type yang:date-and-time;
    description
      "The time at which the RPC was executed.";
  }

  description
    "Adds the time element to <get>.";
}

```

```

augment /nc:copy-config/nc:input {
  leaf scheduled-time {
    type yang:date-and-time;
    description
      "The time at which the RPC is scheduled to be performed.";
  }

  leaf get-time {
    type empty;
    description
      "Indicates that the rpc-reply should include the
      execution-time.";
  }

  description
    "Adds the time element to <copy-config>.";
}

augment /nc:copy-config/nc:output {
  leaf execution-time {
    type yang:date-and-time;
    description
      "The time at which the RPC was executed.";
  }

  description
    "Adds the time element to <copy-config>.";
}

augment /nc:edit-config/nc:input {

```

```

leaf scheduled-time {
    type yang:date-and-time;
    description
        "The time at which the RPC is scheduled to be performed.";
}

leaf get-time {
    type empty;
    description
        "Indicates that the rpc-reply should include the

```

```

        execution-time.";
    }

    description
        "Adds the time element to <edit-config>.";
}

augment /nc:edit-config/nc:output {
    leaf execution-time {
        type yang:date-and-time;
        description
            "The time at which the RPC was executed.";
    }

    description
        "Adds the time element to <edit-config>.";
}

augment /nc:delete-config/nc:input {
    leaf scheduled-time {
        type yang:date-and-time;
        description
            "The time at which the RPC is scheduled to be performed.";
    }

    leaf get-time {
        type empty;
        description
            "Indicates that the rpc-reply should include the
            execution-time.";
    }
}

```

```

    description
      "Adds the time element to <delete-config>.";
  }

  augment /nc:delete-config/nc:output {
    leaf execution-time {
      type yang:date-and-time;
      description
        "The time at which the RPC was executed.";
    }
  }

```

```

  }
  description
    "Adds the time element to <delete-config>.";
}

augment /nc:lock/nc:input {
  leaf scheduled-time {
    type yang:date-and-time;
    description
      "The time at which the RPC is scheduled to be performed.";
  }

  leaf get-time {
    type empty;
    description
      "Indicates that the rpc-reply should include the
      execution-time.";
  }

  description
    "Adds the time element to <lock>.";
}

augment /nc:lock/nc:output {
  leaf execution-time {
    type yang:date-and-time;
    description
      "The time at which the RPC was executed.";
  }

  description
    "Adds the time element to <lock>.";
}

```

```

augment /nc:unlock/nc:input {
  leaf scheduled-time {
    type yang:date-and-time;
    description
      "The time at which the RPC is scheduled to be performed.";
  }
}

```

```

    leaf get-time {
      type empty;
      description
        "Indicates that the rpc-reply should include the
         execution-time.";
    }

    description
      "Adds the time element to <unlock>.";
  }

augment /nc:unlock/nc:output {
  leaf execution-time {
    type yang:date-and-time;
    description
      "The time at which the RPC was executed.";
  }

  description
    "Adds the time element to <unlock>.";
}

augment /nc:close-session/nc:input {
  leaf scheduled-time {
    type yang:date-and-time;
    description
      "The time at which the RPC is scheduled to be performed.";
  }

  leaf get-time {
    type empty;
    description
      "Indicates that the rpc-reply should include the
       execution-time.";
  }
}

```

```

    }

    description
      "Adds the time element to <close-session>.";
  }

  augment /nc:close-session/nc:output {

```

```

    leaf execution-time {
      type yang:date-and-time;
      description
        "The time at which the RPC was executed.";
    }

    description
      "Adds the time element to <close-session>.";
  }

  augment /nc:kill-session/nc:input {
    leaf scheduled-time {
      type yang:date-and-time;
      description
        "The time at which the RPC is scheduled to be performed.";
    }

    leaf get-time {
      type empty;
      description
        "Indicates that the rpc-reply should include the
        execution-time.";
    }

    description
      "Adds the time element to <kill-session>.";
  }

  augment /nc:kill-session/nc:output {
    leaf execution-time {
      type yang:date-and-time;
      description
        "The time at which the RPC was executed.";
    }
  }

```

```

    description
      "Adds the time element to <kill-session>.";
  }

```

```

augment /nc:commit/nc:input {
  leaf scheduled-time {

```

```

    type yang:date-and-time;
    description
      "The time at which the RPC is scheduled to be performed.";
  }

  leaf get-time {
    type empty;
    description
      "Indicates that the rpc-reply should include the
       execution-time.";
  }

  description
    "Adds the time element to <commit>.";
}

augment /nc:commit/nc:output {
  leaf execution-time {
    type yang:date-and-time;
    description
      "The time at which the RPC was executed.";
  }

  description
    "Adds the time element to <commit>.";
}

augment /ncm:netconf-state {
  container scheduling-tolerance {
    uses scheduling-tolerance-parameters;
    description
      "The scheduling tolerance when the time capability
       is enabled.";
  }
  description
    "The scheduling tolerance of the server.";
}

```

```

}

rpc cancel-schedule {
  description
    "Cancels a scheduled message.";

```

```

reference
  "draft-mm-netconf-time-capability:
  Time Capability in NETCONF";

input {
  leaf cancelled-message-id {
    type string;
    description
      "The ID of the message to be cancelled.";
  }
  leaf get-time {
    type empty;
    description
      "Indicates that the rpc-reply should include
      the execution-time.";
  }
}

output {
  leaf execution-time {
    type yang:date-and-time;
    description
      "The time at which the RPC was executed.";
  }
}
}

notification netconf-scheduled-message {
  leaf schedule-id {
    type string;
    description
      "The ID of the scheduled message.";
  }

  leaf scheduled-time {
    type yang:date-and-time;
    description
      "The time at which the RPC is scheduled to be performed.";
  }
}

```

}

description

"Indicates that a scheduled message was received.";

Mizrahi, Moses

Expires November 7, 2015

[Page 29]

Internet-Draft

Time Capability in NETCONF

May 2015

reference

["draft-mm-netconf-time-capability:](#)

Time Capability in NETCONF";

}

}

<CODE ENDS>

Authors' Addresses

Tal Mizrahi

Department of Electrical Engineering

Technion - Israel Institute of Technology

Technion City, Haifa, 32000, Israel

Email: dew@tx.technion.ac.il

Yoram Moses

Department of Electrical Engineering

Technion - Israel Institute of Technology

Technion City, Haifa, 32000, Israel

Email: moses@ee.technion.ac.il

Mizrahi, Moses

Expires November 7, 2015

[Page 30]