Conditional Enablement of Configuration Nodes

draft-kwatsen-conditional-enablement-00

Abstract

This memo presents a cross-cutting technique whereby a NETCONF server can support conditional enablement of configuration nodes. That is, whether the node is active or not depends on the evaluation of an expression. Two expression types are defined herein, one for latent configuration (present but not actualized) and another for temporal configuration (actualized based on time). This solution presented is extensible so that additional expression types may be added in the future.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 22, 2013.

Copyright Notice

Copyright (c) 2013 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. Requirements Terminology ........................................ 3
2. Introduction ....................................................... 3
3. Motivation .................................................................. 3
   3.1. Explicit Nodes Defined in NETMOD Drafts .................. 3
   3.2. Precedent in Juniper's JUNOS-Based Products ............. 4
   3.3. Use-Cases Scoped By I2RS Working Group ................... 5
4. Expression Types ..................................................... 5
   4.1. Overview ......................................................... 5
   4.2. Simple Expressions ............................................. 5
   4.3. Complex Expressions ........................................... 5
5. Feature Types .......................................................... 6
   5.1. Overview ......................................................... 6
   5.2. Simple ............................................................. 6
   5.3. Time ................................................................. 7
6. Conditional Enablement Capability ............................... 7
   6.1. Overview ......................................................... 7
   6.2. Dependencies ................................................... 7
   6.3. Capability Identifier .......................................... 7
   6.4. New Operations ................................................ 7
   6.5. Modifications to Existing Operations ....................... 7
   6.6. Interactions with Other Capabilities ....................... 8
7. Security Considerations ............................................. 8
8. IANA Considerations ................................................. 8
9. Normative References ............................................... 8
1. Requirements Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. Introduction

This memo presents a cross-cutting technique whereby a NETCONF server can support conditional enablement of configuration nodes. That is, whether the node is active or not depends on the evaluation of an expression. Two expression types are defined herein, one for latent configuration (present but not actualized) and another for temporal configuration (actualized based on time). This solution presented is extensible so that additional expression types may be added in the future.

3. Motivation

3.1. Explicit Nodes Defined in NETMOD Drafts

Two separate drafts presented during the NETMOD meeting and IETF 85 had data-models with explicit "enabled" leaves (see examples below). One of the questions asked during the sessions was if cross-cutting concerns, such as if a node were enabled, wouldn't be better supported via meta-data, to which there was general agreement.

Example of an "enabled" leaf from draft-ietf-netmod-routing-cfg-06.txt:

```lxml
leaf enabled {
    type boolean;
    default "true";
    description
        "Enable/disable the router instance.
```
If this parameter is false, the parent router instance is disabled, despite any other configuration that might be present.

Figure 1

Example of an "enabled" leaf from draft-ietf-netmod-system-mgmt-04.txt:

```plaintext
leaf enabled {
    type boolean;
    default true;
    description "Indicates whether this server is enabled for use or not."
}
```

Figure 2

3.2. Precendent in Juniper's JUNOS-Based Products

Further, there is already a precedent for this strategy in Juniper's JUNOS-based products, where any configuration node can be flagged with an XML attribute stating that it is inactive.

Example of a JUNOS "inactive" statement:

```xml
<inteface inactive="inactive">...<interface>
```

Figure 3

Additionally, JUNOS also supports the notion of a list of conditionals that must all be satisfied for an associated configuration to be applied. JUNOS supports the conditionals to be based on chassis type, model type, routing engine, member, and time.
Example of a JUNOS "when" statement (this is not YANG):

```
groups {
  my-group-g1 {
    system {
      hostname xyz;
    }
    when {
      model tx1000;
      routing-engine re0;
      time 2am to 4am;
    }
  }
}
```

Figure 4

3.3. Use-CasesScoped By I2RS Working Group

Lastly, discussions with the I2RS Working Group have revealed that they believe they have a need for a device to autonomously switch configuration settings based on time.

4. Expression Types

4.1. Overview

Conditional expressions are boolean expressions that evaluate to either "true" or "false".

Expressions are contained inside an XML attribute called "enabled". An expression that evaluates to "true" enables the associated node, whereas an expression that evaluates to "false" disables it.

A configuration node having no expression set is equivalent to an expression evaluating to "true". That is, all nodes are enabled by default.

Example usage:
4.2. Simple Expressions

Simple expressions are the most trivial expressions possible; they are simply either the constant value "true" or "false".

\[
\text{expressions} = \text{"true" / "false";}\]

Simple expressions are useful to disable a configuration node until it is explicitly reenabled. Since this is such a common use-case, this draft enables simple expressions to be supported without having to implement support for complex expressions.

A device advertises support for simple expressions using the feature "simple" in its capability string:

\[
\text{<capability string>?features=simple}\]

4.3. Complex Expressions

All expressions that are not "simple" are "complex". These expressions require being able to compare values and evaluate logical expressions; they do not need to perform arithmetic, bitwise operations, or assignments.

The grammar is the same for all complex expressions, the only thing that varies is what variable assignments the device supports (e.g. time, reference to a statistical value, etc.).

A device does not explicitly advertise support for complex expressions. Support for complex expressions are implicit when any feature depending on complex expressions (e.g. time) is advertised. For instance:

\[
\text{<capability string>?features=time}\]

Complex expressions use the following grammar:

\[
\text{<expression> } = \text{ [ <paren-expr> / <and-expr> / <or-expr> / <not-expr> / <static-expr> / <simple-expr> ]}\]
5. Feature Types

5.1. Overview

The types of expressions a device supports is advertised as a list of "features" in the capability identifier string.

5.2. Simple

The "simple" feature defines support for constant boolean values "true" and "false". Simple expressions are useful to disable a node until it is explicitly reenabled.

5.3. Time

The "time" feature defines support for complex expressions using time-oriented values such as "dayofweek" and "hour". Time expressions are useful to implement policies that depend on time.

6. Conditional Enablement Capability
6.1. Overview

The :conditional-enablement capability advertises that the NETCONF server supports the ability for nodes in its to be annotated with metadata specifying conditions when it is enabled or its values vary.

6.2. Dependencies

None.

6.3. Capability Identifier

The :conditional-enablement capability is identified by the following capability string:

```
urn:ietf:params:netconf:capability:conditional-enablement:1.0? \ 
features={name,...}
```

The :conditional-enablement capability URI MUST contain a "features" argument assigned a comma-separated list of names indicating which expression-types the NETCONF peer supports. For example:

```
urn:ietf:params:netconf:capability:conditional-enablement:1.0? \ 
features/simple,time
```

6.4. New Operations

None.

6.5. Modifications to Existing Operations

The :conditional-enablement capability modifies any operation that transmits configuration, including:

```
<get-config>
<edit-config>
<copy-config>
```
that any node in its configuration MAY contain XML attributes defined in the "Overview" section of this capability, even though those attributes were not explicitly defined in its YANG module.

6.6. Interactions with Other Capabilities

None.

7. Security Considerations

There are no known security considerations at this time.

8. IANA Considerations

There are no IANA directives.

9. Normative References


Author's Address

Kent Watsen
Juniper Networks

EMail: kwatsen@juniper.net