Network Modeling
Internet-Draft
Intended status: Standards Track
Expires: 27 June 2026

R. Wills, Ed.
Cisco
Q. Ma
Huawei
D. Rajaram
Nokia

24 December 2025

YANG Configuration Templates
draft-tt-netmod-yang-config-templates-01

Abstract

NETCONF and RESTCONF protocols provide programmatic interfaces for accessing configuration data modeled by YANG. This document defines the use of a YANG-based configuration template mechanism whereby configuration data can be defined in one or more templates and applied repeatedly. This avoids the redundant definition of identical configuration and ensures the consistency of it, thus allowing devices to be managed more conveniently and efficiently.

Status of This Memo

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

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

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

This Internet-Draft will expire on 27 June 2026.

Table of Contents

1. Introduction

This document considers the case of a datastore that contains multiple subtrees with similar or identical nodes within them, such that the datastore contains repetitive data with limited variation. If a client has to repeatedly configure the same nodes for each subtree, this can become complex, error-prone, and masks the intent of the client.

This document proposes a solution to improve this, called "Configuration Templates", that results in a smaller running datastore even when the configuration in <running> is large.

A Configuration Template is a fragment of configuration that the device is instructed to replicate multiple times to generate copies of the configuration. This allows repetitive subtrees of configuration to be written only once, in the template. When needed, individual instantiations of a template can override the values of nodes, or add new instance-specific nodes.

NMDA [RFC8342] allows the configuration templates to be defined in <running> and expanded in <intended>, but it does not specify details about how configuration templates could be created and applied.

This document defines the use of configuration templates in the context of YANG-driven network management protocols such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Configuration templates can be used with any YANG data model, this document doesn’t make any assumption on the YANG data model design, i.e. it does not rely on a shared profile/group being defined in the YANG data model.

1.1. Editorial Note (To be removed by RFC Editor)

Note to the RFC Editor: This section is to be removed prior to publication.

This document contains placeholder values that need to be replaced with finalized values at the time of publication. This note summarizes all of the substitutions that are needed. No other RFC Editor instructions are specified elsewhere in this document.

Please apply the following replacements:

* XXXX --> the assigned RFC number for this draft

* 2025-05-28 --> the actual date of the publication of this document

2. Conventions and Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The meanings of the symbols in tree diagrams are defined in [RFC8340].

This document uses the YANG terminology defined in Section 3 of [RFC7950].

Besides, this document defines the following terminology:

Configuration Template: A chunk of reusable configuration data that could be applied to the configuration repeatedly, in order to simplify the delivery of network configuration and ensure the consistency of it. A configuration template may also be called "template" or "YANG template" throughout this document.

3. Requirements

This section describes the requirements that the Configuration Templates solution must satisfy. These requirements were all discussed in the Interim Meetings, and a rough consensus was reached on each of them by the participants in the meetings. A general theme of the Configuration Templates work is to come up with a "Minimal Viable Product" that is useful but not over-complicated. More advanced features could be considered as extensions in later drafts.

3.1. Defining and Managing Templates

Templates can be used with any YANG module. They contain nodes of configuration data, and are stored persistently in the running datastore of the device.

A client can view and manipulate a template, including the configuration inside it, by manipulating it in the <running> datastore. In this sense, a template and its contents behaves like any other subtree of configuration.

3.2. Applying Templates

A template can be applied to zero or more nodes in the <running> datastore. Each node can have zero or more templates applied to it, and the order they are applied is specified by the client. The order is important when determining the final intended configuration -- see the next section.

Templates can be applied at multiple points in the hierachy. The next section states the requirements when a node applies a template and it has an ancestor that also applies a template.

When viewing the <running> datastore, there is a mechanism to see which templates have been applied to each node, and in which order.

3.3. Producing the Intended Datastore

The device's <intended> datastore is the result of combining all the applications of templates together with non-template config. This is called "expanding out" the templates.

The intended configuration inside a subtree is the result of taking the relevant contents of every template applied to the subtree's root node and its ancestors, and combining it with the (non-template) data nodes inside the subtree.

A node inside a subtree may be present in multiple templates that have been applied, and/or it may be present as non-template config inside the subtree. The requirements for combining the templates and the non-template config together are as follows:

* The value of a node in the <intended> configuration is determined by using precedence to decide where to take the value from.

* Non-template config always has the highest precedence.

* When templates are applied to multiple ancestors, the innermost ancestor takes precedence.

* When multiple templates are applied to a particular node, the order of application (as indicated by the client when applying the templates) determines the precedence within that node.

Whenever the contents of a template is updated in <running>, the result of expanding out the template appears in <intended> and takes effect on the device.

3.4. Pattern Matching in Templates

The configuration inside a template definition can contain values for list keys that are simple regular expressions, using a limited subset of regular expression syntax. This controls which list entries that particular subtree of the template takes effect for when the template is applied.

An example of this would be to have a template that is applied to a top-level "interfaces" container, but the template only takes effect for certain interface names that match the regular expression.

3.5. Off-box Template Expansion

If the client knows the contents of the <running> datastore (non-template config, template definitions and template applications), it must be possible for the client to calculate the result of template expansion.

In other words, the outcome of template expansion depends solely on the <running> datastore and not the state of the device.

4. Configuration Template Solution

4.1. Defining Templates

A configuration template must first be defined before it can be applied (see Section 4.2). The creation, modification, and deletion of configuration templates is achieved by network management operations via NETCONF or RESTCONF protocols. The contents of the configuration template must be an instantiated chunk of data starting from any level node in the module hierarchies.

(Editor’s note: more work may be needed here to ensure the template is a valid subtree of config from a schema perspective. This may mean we need a way of saying where the root of the template is in the schema, for example with a set of "outer" nodes with operation="none").

The YANG data model of configuration templates is defined in Section 7.

4.1.1. Templates with Regular Expressions

Simple regular expressions can be used to restrict which list entries a template takes effect for.

(Editor's note: more work is needed here to define the exact semantics of this. Also, the regular expressions will be very simple (again, this needs to be defined), and therefore it may be better to call them 'globs' or 'patterns')

For example, Figure 1 provides an interface configuration template that sets "type" as ethernetCsmacd and "mtu" as 1500 for interfaces named with the prefix "eth":

```
<templates xmlns="urn:ietf:params:xml:ns:yang:ietf-config-template">
  <template>
    <id>ethernet-interface</id>
    <content>
      <interfaces xmlns="urn:example:interface">
        <interface>
          <name>^eth.*</name>
          <type>ethernetCsmacd</type>
          <mtu>1500</mtu>
        </interface>
      </interfaces>
    </content>
  </template>
</templates>
```

Figure 1: Example of An Interface template

4.2. Applying Templates

For each configuration node in the <running> datastore, one or more templates can be applied. This causes configuration from the templates to be combined with child configuration in the <running> datastore to produce a final set of <intended> configuration that will be used by the device.

4.2.1. The "apply-templates" Metadata

Template application is indicated using the "apply-templates" metadata. The value of this is a list of space-separated template identifiers. If the template is applied to a node in the data tree, the metadata object is added to that specific node.

The encoding of "apply-templates" metadata object follows the way defined in Section 5 of [RFC7952].

For example, the following interface configuration may be provided with the container node "interfaces" applying the template defined in Figure 1:

```
<interfaces xmlns="urn:example:interface"
  xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-config-template"
  ct:apply-templates="ethernet-interface">
  <interface>
    <name>loopback0</name>
  </interface>
  <interface>
    <name>eth0</name>
  </interface>
  <interface>
    <name>eth1</name>
  </interface>
</interfaces>
```

And the above interface configuration renders the following expanded configuration:

```
<interfaces xmlns="urn:example:interface">
  <interface>
    <name>loopback0</name>
  </interface>
  <interface>
    <name>eth0</name>
    <type>ethernetCsmacd</type>
    <mtu>1500</mtu>
  </interface>
  <interface>
    <name>eth1</name>
    <type>ethernetCsmacd</type>
    <mtu>1500</mtu>
  </interface>
</interfaces>
```

4.2.2. Creating, editing and deleting the "apply-templates" metadata

The apply-templates metadata can be modified by the client by specifying it as an attribute in an <edit-config> request. There are three cases:

* The apply-templates attribute is specified and the value is non-empty (i.e. a list of templates to apply to the node). The apply-templates metadata is changed to match the value in the request.

* The apply-templates attribute is specified and the value is the empty string. The apply-templates metadata is removed and thus no templates are applied to the node.

* The apply-templates attribute not specified. The apply-templates metadata currently present on the node (if any) is unchanged.

For example, this request creates a single loopback0 interface and applies template t1 to the interfaces container:

```
<edit-config>
  ...
  <config>
    <interfaces xmlns="urn:example:interface"
                ct:apply-templates="t1">
      <interface>
        <name>loopback0</name>
      </interface>
    </interfaces>
  </config>
</edit-config>
```

This request also applies template t2 to the interfaces container:

```
<edit-config>
  ...
  <config>
    <interfaces xmlns="urn:example:interface"
                ct:apply-templates="t1 t2" />
  </config>
</edit-config>
```

After this request, <running> is as follows:

```
<interfaces xmlns="urn:example:interface"
            ct:apply-templates="t1 t1">
  <interface>
    <name>loopback0</name>
  </interface>
</interfaces>
```

This request adds a new interface list entry, and leaves the applied templates unchanged:

```
      <edit-config>
        ...
        <config>
          <interfaces xmlns="urn:example:interface">
            <interface>
              <name>eth0</name>
            </interface>
          </interfaces>
        </config>
      </edit-config>
```

After this request, <running> is as follows:

```
      <interfaces xmlns="urn:example:interface"
                  ct:apply-templates="t1 t1">
        <interface>
          <name>loopback0</name>
        </interface>
        <interface>
          <name>eth0</name>
        </interface>
      </interfaces>
```

Finally, this request deletes all the templates, and leaves the list entries unchanged:

```
      <edit-config>
        ...
        <config>
          <interfaces xmlns="urn:example:interface"
                      ct:apply-templates="" />
          </interfaces>
        </config>
      </edit-config>
```

After this request, <running> is as follows:

```
      <interfaces xmlns="urn:example:interface">
        <interface>
          <name>loopback0</name>
        </interface>
        <interface>
          <name>eth0</name>
        </interface>
      </interfaces>
```

4.3. Overriding Templates

The client may want to to override some configuration in a template when it is applied to a particular node in <running>. The client can achieve this by providing the desired value at the corresponding level when applying the template. Configuration explicitly provided by the client always takes precedence over the same node defined in template.

A template node can be overriden by having its value changed, but it can’t be deleted.

As an example of overriding a node in a template, a client may configure physically present interfaces "eth0" and "eth1" inheriting the template defined in Figure 1, but the "mtu" value of "eth1" needs to be 9122:

```
<interfaces xmlns="urn:example:interface"
  xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-config-template"
  ct:apply-templates="ethernet-interface">
  <interface>
    <name>loopback0</name>
  </interface>
  <interface>
    <name>eth0</name>
  </interface>
  <interface>
    <name>eth1</name>
    <mtu>9122</mtu>
  </interface>
</interfaces>
```

And the above interface configuration renders the following expanded configuration:

```
<interfaces xmlns="urn:example:interface">
  <interface>
    <name>loopback0</name>
  </interface>
  <interface>
    <name>eth0</name>
    <type>ethernetCsmacd</type>
    <mtu>1500</mtu>
  </interface>
  <interface>
    <name>eth1</name>
    <type>ethernetCsmacd</type>
    <mtu>9122</mtu>
  </interface>
</interfaces>
```

4.4. Expanding Templates

When a configuration template is applied to a node in the data tree, it acts as if the configuration defined in the template is merged with the configuration provided explicitly at the corresponding level in the data tree, with the explicitly provided configuration taking precedence.

The rules for deriving the <running> configuration are as follows:

* The value of a node in the <intended> configuration is determined by using precedence to decide where to take the value from.

* Non-template config always has the highest precedence.

* When templates are applied to multiple ancestors, the innermost ancestor takes precedence.

* When multiple templates are applied to a particular node, the order of application (as indicated by the client when applying the templates) determines the precedence within that node.

Whenever the contents of a template is updated in <running>, the result of expanding out the template appears in <intended> and takes effect on the device.

4.5. Validity of Templates

The contents of the template alone is not always sufficient to enforce the constraints of the data model. Some constraints may depend on configuration outside of the templates to satisfy, e.g., a list may contain a mandatory leaf node which is not defined in the template but explicitly provided by the client. However, servers SHOULD parse the template and enforce the constraints if it is possible during the processing of template creation, e.g., servers may validate type constraints for the leaf, including those defined in the type's "range", "length", and "pattern" properties.

That said, if a template is applied in the configuration data tree, the results of the template configuration merging with configuration explicitly provided by the client MUST always be valid, as defined in Section 8.1 of [RFC7950].

5. Interaction with NMDA datastores

Some implementations may have predefined configuration templates for the convenience of clients, which are present in <system> (if implemented, see [I-D.ietf-netmod-system-config]). In addition, clients can always define their own templates in <running>. However, configuration template data defined by "ietf-config-template" YANG data model should not be visible in <operational> until being inherited by a node in the data tree.

If a node in the data tree applies a configuration template, the configuration template does not expand in <running>. A read of <running> returns what is sent by the client with the "apply-templates" metadata attached to the specific node. A configuration template which is inherited or overridden by the node instance MUST be expanded in <intended>.

6. Interaction with Non-NMDA datastores

TBC

7. The "ietf-config-template" YANG Module

7.1. Data Model Overview

The following tree diagram [RFC8340] illustrates the "ietf-config-template" module:

```
module: ietf-config-template
  +--rw templates
     +--rw template* [id]
        +--rw id               string
        +--rw description?     string
        +--rw content?         <anydata>
        +--ro last-modified?   yang:timestamp
```

Editor's Note: Should we use the RFC7952 metadata annotation for the 'apply-templates' metadata here?

Editor's Note: the current definition of template configuration uses anydata, but this may not be able to be validated at template definition time because anydata is opaque.

7.2. YANG Module

```
<CODE BEGINS> file "ietf-template@2025-05-28.yang"
module ietf-config-template {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-config-template";
  prefix ct;

  import ietf-yang-types {
    prefix yang;
    reference
      "RFC 6991: Common YANG Data Types";
  }

  organization
    "IETF NETMOD (Network Modeling) Working Group";
  contact
    "WG Web:   <https://datatracker.ietf.org/wg/netmod/>
     WG List:  <mailto:netmod@ietf.org>
     Author: Qiufang Ma
             <mailto:maqiufang1@huawei.com>";

  description
    "This module defines a template list with a RPC to expand
     the template.

     Copyright (c) 2025 IETF Trust and the persons identified
     as authors of the code. All rights reserved.

     Redistribution and use in source and binary forms, with
     or without modification, is permitted pursuant to, and
     subject to the license terms contained in, the Revised
     BSD License set forth in Section 4.c of the IETF Trust's
```

```
        Legal Provisions Relating to IETF Documents
        (https://trustee.ietf.org/license-info).

        This version of this YANG module is part of RFC XXXX
        (https://www.rfc-editor.org/info/rfcXXXX); see the RFC
        itself for full legal notices.

        The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL',
        'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED',
        'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this document
        are to be interpreted as described in BCP 14 (RFC 2119)
        (RFC 8174) when, and only when, they appear in all
        capitals, as shown here.";

      revision 2025-05-28 {
        description
          "Initial revision.";
        reference
          "RFC XXXX: YANG Templates";
      }

      container templates {
        description
          "Specifies the template parameters.";
        list template {
          key id;
          description
            "The list of templates managed on this device.";
          leaf id {
            type string;
            description
              "The identifier of the template that uniquely identifies a
               template.";
          }
          leaf description {
            type string;
            description
              "A textual description of the template.";
          }
          anydata content {
            description
              "inline template content.";
          }
          leaf last-modified {
            type yang:timestamp;
            config false;
            description
              "Timestamp when the template is modified last time.";
```

```
          }
        }
      }
   }
   <CODE ENDS>
```

8. Security Considerations

TODO Security

9. IANA Considerations

9.1. The "IETF XML" Registry

This document registers the following URI in the "IETF XML Registry" [RFC3688].

```
        URI: urn:ietf:params:xml:ns:yang:ietf-config-template
        Registrant Contact: The IESG.
        XML: N/A, the requested URI is an XML namespace.
```

9.2. The "YANG Module Names" Registry

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

```
     name:               ietf-config-template
     namespace:          urn:ietf:params:xml:ns:yang:ietf-config-template
     prefix:             ct
     maintained by IANA? N
     reference:          RFC XXXX
```

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/rfc/rfc3688>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/rfc/rfc6020>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/rfc/rfc6241>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/rfc/rfc7950>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/rfc/rfc8040>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

10.2. Informative References

[I-D.ietf-netmod-system-config] Ma, Q., Wu, Q., and C. Feng, "System-defined Configuration", Work in Progress, Internet-Draft, draft-ietf-netmod-system-config-15, 12 February 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-netmod-system-config-15>.

[RFC7952] Lhotka, L., "Defining and Using Metadata with YANG", RFC 7952, DOI 10.17487/RFC7952, August 2016, <https://www.rfc-editor.org/rfc/rfc7952>.

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

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/rfc/rfc8342>.

Acknowledgments

The author would like to thank Lou Berger, Jason Sterne, Kent Watsen, and Robert Wilton for comments and contributions made during interim meetings.

The author would like to acknowledge the following drafts and presenters for kick-starting discussions on Yang Templates:

* draft-ma-netmod-yang-config-template-00

* draft-rajaram-netmod-yang-cfg-template-framework-00

* draft-wills-netmod-yang-templates-00

* Jan Lindblad

Contributors

Qin Wu
Huawei
101 Software Avenue, Yuhua District
Jiangsu
210012
China
Email: bill.wu@huawei.com

Authors' Addresses

Robert Wills (editor)
Cisco
United Kingdom
Email: rowills@cisco.com

Qiufang Ma
Huawei
101 Software Avenue, Yuhua District
Jiangsu
210012
China
Email: maqiufang1@huawei.com

Deepak Rajaram
Nokia
India

Email: deepak.rajaram@nokia.com