Network Modeling
Internet-Draft
Intended status: Informational
Expires: 19 March 2026

C. Cardona
NTT
B. Claise
Everything OPS

15 September 2025

Guidelines for YANG Example Validation and Coverage Analysis in IETF Documents
draft-cardona-claise-onion-yang-coverage-00

Abstract

This document defines guidelines for including YANG example instances in IETF documents in a way that enables automatic extraction and validation. It introduces the concept of YANG module "coverage" to measure how thoroughly example instances exercise a YANG module's data nodes. The goal is to improve the quality and usability of YANG models in IETF documents by providing authors with tools to validate their examples and assess coverage completeness.

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 19 March 2026.

Copyright Notice

Table of Contents

1. Introduction 2
2. Including YANG Example Instances in IETF Documents 3
2.1. Types of Example Validation 4
2.1.1. Simple Validation Examples 5
2.1.2. Complex Examples with External References 5
2.2. Including Auxiliary YANG Modules for Examples 6
3. YANG Module Coverage by Examples 7
3.1. Coverage Example 8
3.2. Coverage Across Multiple Documents 9
4. Implementation 10
5. Conclusion 10
6. IANA Considerations 10
7. Security Considerations 11
8. References 11
8.1. Normative References 11
8.2. Informative References 11
Authors' Addresses 12

1. Introduction

Although the YANG data modeling language has been in use for over a decade, there remains a lack of tooling to help YANG model authors analyze and validate their YANG modules and example instances. People defining or evaluating YANG modules have limited guidance on how to include and test example data for those modules. Example data instances are vital to illustrate how a model is intended to be used, but the guidelines for presenting and validating these examples are scarce. The updated guidelines in RFC 8407bis improve the situation, but gaps still exist in providing tools that can automatically give authors feedback on what they might be missing in their models.

The objective of this document is to define guidelines that enable the development of tools which, given one or more IETF documents (RFCs or drafts), can verify that the YANG model examples in those documents are correct and sufficiently comprehensive. To achieve this, we address two main aspects:

* Guidelines for Including YANG Examples in RFCs: A clear method for authors to include example data instances of YANG models in IETF documents in a way that tools can automatically extract and validate them. In addition, we outline how tools can perform this extraction and validation, and we propose a method to include extra YANG modules (when necessary) for validating more complex examples without affecting the main normative content.

* Definition of YANG Module "Coverage": A new concept to measure how much of a YANG module's data nodes are exercised by the example instances in a document. By analyzing the examples, a tool can inform authors which parts of their YANG model are covered and which are still missing from the examples. A proof-of-concept implementation for computing such coverage is also considered.

Scope: The proposals in this document focus on typical use cases. It is expected that proof-of-concept tools will confirm the viability of these methods for moderately complex examples as this document progresses. If such automatic validation proves infeasible in practice, the approach may need adjustment. Extremely complex example scenarios (e.g., those involving schema mount or highly dynamic state) are outside the current scope and could be addressed in future work.

2. Including YANG Example Instances in IETF Documents

When authors include example YANG instances in an RFC or Internet-Draft, those example instances SHOULD be formatted in a way that allows automatic extraction and validation by tools.

Specifically, example instances SHOULD be enclosed as code blocks using the CODE BEGINS / CODE ENDS convention (as defined in the updated guidelines, e.g., draft-ietf-netmod-rfc8407bis). They should also be given a file name that clearly identifies them as examples, analogous to how YANG modules are named (see RFC 8407bis guidelines). We propose that the file name ends with the suffix .instance.xml for XML instances or .instance.json for JSON instances. This naming convention makes it easy for tools to identify which code snippets are YANG instance examples that can be parsed and validated.

Each example file name SHOULD be unique within a document (and ideally unique across related documents being analyzed together). Tools may complain or error out if they encounter duplicate example file names, since that could indicate conflicting or repeated content. For example, an XML instance might be included as:

```
// Not an actual instance
<CODE BEGINS> file "interface-config.instance.xml"
<?xml version="1.0" encoding="UTF-8"?>
<data> ... </data>
<CODE ENDS>
```

And a JSON instance as:

```
// Not an actual instance
<CODE BEGINS> file "interface-config.instance.json"
{
  "interface": { ... }
}
<CODE ENDS>
```

By following this approach, a tool can scan the document for code blocks with filenames ending in .instance.xml or .instance.json, extract their contents, and validate them against the appropriate YANG modules.

In cases where an included code snippet is not meant to represent a valid YANG instance (for instance, it might be pseudocode, a partial fragment, or intentionally invalid to illustrate an error), the author can indicate this so that automated validators skip it. For example, an author might choose not to use the CODE BEGINS/CODE ENDS markers or might omit the special .instance.xml/.json filename suffix for such a snippet. Without the markers or filename, the tool will know not to treat that snippet as a YANG instance example to be validated. An author could also include an explicit note in the text indicating that a given snippet is not for validation. Using these conventions strikes a balance between making examples machine-checkable and allowing flexibility for authors to include non-valid illustrative snippets when necessary.

2.1. Types of Example Validation

Not all example instances are the same in terms of validation. Many YANG examples in documentation are simple and only involve the module defined in the document, but in more complex scenarios an example might span multiple modules or need knowledge of deviations, features, or other context. For the latter, wrapping the example in an instance data file format (defined by RFC 9195) can make explicit which modules and context are in use. This approach can handle more complicated example validation at the cost of a bit more verbosity. (If a given example scenario is too complex to easily capture even with the instance-data file format - for example, requiring extensive setup or dynamic state - it is considered out of scope for automatic validation in this document.) We distinguish between two categories

of examples for validation purposes:

2.1.1. Simple Validation Examples

These are straightforward examples that instantiate YANG modules defined in the same document, with a self-contained context. An extraction tool can load the YANG module(s) defined in the document (plus any standard modules they import) and then validate the example instance against those modules. For instance, if a draft defines a YANG module example-interfaces and provides a small XML snippet configuring an interface, the tool will use the example-interfaces module (and any of its dependencies) to verify that the XML conforms to the schema. In general, simple examples illustrate how to use the module(s) in the draft and can be validated with minimal additional information beyond what the document itself provides.

2.1.2. Complex Examples with External References

These are examples that require additional context or external modules beyond those defined in the current document. For example, an instance data set may involve multiple modules (perhaps a module from this draft plus some from other RFCs), or it might describe operational state data that requires a specific datastore context or feature set. For such cases, we propose that the example be presented as a self-contained YANG instance data file using the standard format defined in RFC 9195 (the YANG Instance Data File Format). In practice, this means the example should be wrapped in the instance-data-set structure which includes the necessary metadata (such as module names, revisions, feature toggle states, and target datastore) for validation.

For an XML-formatted complex example, the first element of the snippet should be <instance-data-set xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-instance-data">, with the content formatted according to RFC 9195. Inside this element, sub-elements specify what YANG content schema the instance data conforms to (including module names, revision dates, features, deviations, and the intended datastore). Similarly, for a JSON-formatted complex example, the top-level JSON object should have a single key "ietf-yang-instance-data:instance-data-set", whose value is an object providing the same kind of metadata and the actual data set. By using the RFC 9195 instance-data-set wrapper, all the information needed to validate the instance is included within the text itself. A validation tool like yanglint (with appropriate support for RFC 9195) can read the instance-data-set, automatically load the indicated modules and their revisions, apply any stated deviations or feature enables, and then verify the embedded data content against the assembled schema. Note that if a tool detects

that an example instance uses the RFC 9195 instance data format, it will attempt to validate it accordingly - and the example instance will fail validation if it does not conform to the declared schema or constraints.

2.2. Including Auxiliary YANG Modules for Examples

Sometimes an example instance cannot be validated or understood without additional YANG definitions that are not part of the main module in the document. A common case is the use of YANG deviations or vendor-specific augmentations to adapt a base model. In principle, deviations are meant to be extraordinary measures (and their use is discouraged in final standards), but in practice they are sometimes used in industry deployments.

To illustrate how a deviation or augmentation affects the data, an RFC might need to include a small auxiliary YANG module (for example, a module that contains deviation statements targeting the base model) purely for use in the example validation. The challenge is to do this without confusing readers or tools and without "contaminating" the main normative content of the document. We propose the following guidelines for including such auxiliary YANG modules in examples:

* Use a Distinct File Extension: Include the auxiliary module as a code block with a filename that uses a different extension than ".yang". For example, use a suffix like .dyang for a deviation/ augmentation module that is only for the example. Using a non-standard extension ensures that general YANG module extraction tools will not treat it as a normative YANG module from the document. Any example that requires this auxiliary module can then reference it by its filename (with the .dyang extension) when running the validation tool.

* Clearly Identify the Module as Non-Normative: The auxiliary module should be clearly identified as an example-only, non-normative module. This can be achieved through both its naming and its content. For instance, the CODE BEGINS label could name it something like "example-device-deviations.dyang" (including words like "example" or "dev" in the filename to signal its purpose). In the module source itself, use a module name and namespace that indicate it is not an official module. For example, use a namespace URI containing "example.com" or similar, and perhaps a module name ending in -example or with a clearly indicative name.

* Explain in the Narrative: In the text surrounding the code, explicitly state that this module is an auxiliary construct provided for the sake of the example. For instance: "The following YANG module is an example showing how a deviation could

be used for this scenario; it is not a normative part of this specification." This clarification is important for human readers and reviewers to avoid any confusion.

By following these practices, authors can include necessary YANG deviations or supplemental modules to support their examples while making it clear that these are not part of the official data model being defined. Tools that extract and compile YANG modules from the document can be instructed to either ignore these example-only modules or handle them separately. For example, a tool might compile them only in the context of validating the corresponding example, and not raise an error about their presence or treat them as missing normative modules in the draft.

3. YANG Module Coverage by Examples

Providing examples in YANG model documents is not just helpful for readers, but can also improve the quality of the model itself. We introduce the concept of YANG module coverage as a way to measure how thoroughly the provided examples exercise the data model.

By analogy to code coverage in software testing (which checks what fraction of code is executed by tests), YANG module coverage looks at what fraction of a module’s schema nodes are present in at least one of the example instances. More specifically, we can define coverage as the set of data nodes (particularly leaf nodes) from the YANG module that have a value in one or more example instances. A simple quantitative metric could be the percentage of the module’s leaf nodes that appear in at least one example.

For instance, if a YANG module defines 100 configurable or state leaf nodes, and the examples collectively include 45 of those leaves, one could say the example coverage is 45%. This number, while not telling the whole story, gives an indication of how much of the model’s functionality is demonstrated by the examples. Low coverage might suggest that important parts of the model are never shown in any example. This could mean those parts are either less relevant or simply that the document might benefit from additional examples to cover them.

Probably more helpful than the coverage percentage, we can also define a "non-coverage tree" - essentially a view of the data model highlighting the nodes that were not covered by any example. This is perhaps the most direct way of showing the authors which portions of their model remain unused in the examples. For example, a tool could output a tree or list of all leaf nodes not present in any example instance, indicating potential gaps in example coverage.

This kind of feedback can help authors identify gaps and improve their documents. If a particular feature or branch of the model has no example, the authors might choose to add one for completeness or clarity. In some cases, it might reveal that certain parts of the model are hard to exemplify or not yet implemented, which could prompt re-thinking that part of the design or at least adding an explanation for why it's not covered by any example.

Coverage works directly on the data nodes from the YANG module. It should be as simple as possible, therefore, we will consider that any data leaf under a list is covered if there is at least an example with an instance of the list that includes the leaf. That is, for a list with 2 different leafs, one leaf covered under one key and another under another one will still yield a 100% coverage of the list.

3.1. Coverage Example

Consider this simple YANG module:

```
module example-device {
  namespace "http://example.com/device";
  prefix "dev";

  container device {
    leaf hostname { type string; }
    leaf location { type string; }
    leaf admin-state { type boolean; }

    list interface {
      key "name";
      leaf name { type string; }
      leaf enabled { type boolean; }
      leaf other { type boolean; }
    }
  }
}
```

And this example instance:

```
<CODE BEGINS> file "device-config.instance.xml"
<device xmlns="http://example.com/device">
  <hostname>router1</hostname>
  <interface>
    <name>eth0</name>
    <enabled>true</enabled>
  </interface>
</device>
<CODE ENDS>
```

Coverage analysis shows:

- Total data nodes: 6 (hostname, location, admin-state, interface/name, interface/enabled, interface/other)

- Covered nodes: 3 (hostname, interface/name, interface/enabled)

- Coverage percentage: 50%

Non-coverage tree (missing nodes):

```
device
├─── location
├─── admin-state
└─── interface
    └─── other
```

This analysis helps authors identify that the 'location' and 'admin-state' leafs need examples.

3.2. Coverage Across Multiple Documents

It is common that a YANG module defined in one document might be used, extended, or exemplified in another document. Examples could be spread across multiple RFCs or drafts (for instance, a base YANG module in an RFC and a usage guide in a separate draft that provides additional examples). While our primary focus is on coverage within a single document's scope (i.e. how well the examples in this document cover this document's module), one could imagine tooling that aggregates examples from multiple sources for a more holistic coverage analysis. For example, if module A is defined in RFC X, and RFC Y (perhaps a companion or a applicability statement) provides additional examples for module A, a combined coverage analysis across X and Y would give a better picture of how module A is covered in total. We will strive to define approaches that enable tooling to do this kind of multi-document coverage assessment automatically in the future.

For the concept of YANG packages [I-D.ietf-netmod-yang-packages], the coverage across the entire package might be of interest.

For the to-be-created ONIONS Working Group, the coverage across (service) abstractions might be of interest.

4. Implementation

TODO. The progress of this document depends on the correct operation of its proposal in cases of relative complexity using an actual implementation.

5. Conclusion

By standardizing how YANG examples are included and validated, we make it easier for implementers and reviewers to trust that the examples in a specification are correct and that a sufficient range of scenarios is covered. The concept of YANG module coverage, while exploratory, can further encourage authors to include more comprehensive examples and help pinpoint which parts of a data model might need additional illustration. Together, these measures aim to improve the overall quality and usability of YANG models in IETF documents.

Ultimately, the success of these proposals will depend on tool support. As a next step, developing proof-of-concept tools or integrations (for example, enhancements to existing YANG validators like yanglint, or new scripts to extract and analyze examples) will be crucial. If moderately complex examples can be automatically extracted, validated, and measured for coverage with such tools, it will validate the approach. On the other hand, if significant obstacles arise during implementation, the recommendations in this document may need to be refined. Our hope is that these initial guidelines spark discussion and experimentation, leading to more robust support for YANG examples and better validation practices in the near future.

6. IANA Considerations

This document has no IANA actions.

7. Security Considerations

This document defines guidelines for including and validating YANG examples in IETF documents. The recommendations do not introduce new security risks. However, implementers of validation tools should be aware of potential security considerations when processing example data from untrusted sources, such as denial of service attacks through maliciously crafted large or complex examples.

8. References

8.1. Normative References

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

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

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

[RFC9195] Lengyel, B. and B. Claise, "A File Format for YANG Instance Data", RFC 9195, DOI 10.17487/RFC9195, February 2022, <https://www.rfc-editor.org/info/rfc9195>.

8.2. Informative References

[I-D.ietf-netmod-rfc8407bis] Bierman, A., Boucadair, M., and Q. Wu, "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", Work in Progress, Internet-Draft, draft-ietf-netmod-rfc8407bis-28, 5 June 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-netmod-rfc8407bis-28>.

[I-D.ietf-netmod-yang-packages] Wilton, R., Rahman, R., Clarke, J., and J. Sterne, "YANG Packages", Work in Progress, Internet-Draft, draft-ietf-netmod-yang-packages-06, 7 July 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-netmod-yang-packages-06>.

[RFC8407] Bierman, A., "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", BCP 216, RFC 8407, DOI 10.17487/RFC8407, October 2018, <https://www.rfc-editor.org/info/rfc8407>.

Authors' Addresses

Camilo Cardona
NTT
164-168, Carrer de Numancia
08029 Barcelona Catalonia
Spain
Email: camilo@gin.ntt.net

Benoit Claise
Everything OPS
Email: benoit@everything-ops.net