Network Modeling
Internet-Draft
Updates: 7950, 7951, 9254 (if approved)
Intended status: Standards Track
Expires: 3 September 2026

M. Matejka
CZ.NIC

2 March 2026

Transcoding Data Modeled with YANG
draft-marenamat-netmod-core-yang-transcoding-00

Abstract

YANG (RFC 7950) is a standardized data modeling language. The data may be encoded in XML (RFC 7950), JSON (RFC 7951), plain CBOR (RFC 9254) or CBOR with standins (draft-bormann-cbor-yang-standin).

This document specifies how to convert data modeled by YANG encoded in one of the formats into another format. Use cases include e.g. local transcoding between Netconf and Restconf for tool compatibility, or reverse proxying while composing multiple backends.

This document also defines a data annotation for additional value type specification. That data annotation behavior updates RFC 7950, RFC 7951 and RFC 9254.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://marenamat.github.io/ietf-draft-marenamat-netmod-core-yang-transcoding/draft-marenamat-netmod-core-yang-transcoding.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-marenamat-netmod-core-yang-transcoding/.

Discussion of this document takes place on the Network Modeling Working Group mailing list (mailto:netmod@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/netmod/. Subscribe at https://www.ietf.org/mailman/listinfo/netmod/.

Source for this draft and an issue tracker can be found at https://github.com/marenamat/ietf-draft-marenamat-netmod-core-yang-transcoding.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Current documents focus on encoding and specifying data for a simple round-trip between a source encoder and a destination decoder. While Section 1.4 of [RFC8040] at least touches the topic of RESTCONF and NETCONF ([RFC6241]) coexisting at one device, there has not yet been any specification for chaining multiple encodings.

At the time of this document, there are at least four kinds of data encoding (XML, JSON, plain CBOR and standin CBOR) and at least three protocol variants (NETCONF, RESTCONF and CORECONF). Implementing all of these may be too costly, and therefore users will tend to deploy convertor tools and proxies to align the otherwise incompatible data and protocols.

One major usecase for this specification is a simple proxy to bridge clients and servers implementing incompatible encodings. Another example usecase is a reverse proxy used as a single endpoint for a device with multiple internal endpoint. Such a proxy may be transparent when the user knowingly deployed it, or opaque if it’s a part of a third party device. The user may even be completely unaware of such a proxy existing.

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.

TODO: define terms like Transcoder, Composer or XML Round Trip.

3. Transcoding Data Formats

Encoding capabilities of XML, JSON or CBOR are quite different. While all values in XML are strings, JSON knows also booleans and integers, and CBOR may even use the tag features to encode various additional types, like time values or IP addresses, as specified in [I-D.bormann-cbor-yang-standin]. This effectively means that a transcoding node either loses type information, or it must infer that information from the data representation.

Generally, a transcoder MUST interpret the incoming data according to the appropriate YANG model, and choose the appropriate output representation for each value according to the model. Transcoders also SHOULD validate the data and issue appropriate error messages to the user. Otherwise, the user may get confused by an error message proxied from the endpoint, referring to a different data format than the user is using.

Transcoders SHOULD quietly canonicalize value representations according to the YANG model. Since by Section 9.1 of [RFC7950] most types have a canonical representation which MUST be used for all conceivable purposes, it’s expected that no valuable information is lost by such conversion.

However, transcoders MUST NOT perform any value conversion which loses data precision.

Examples:

For leaf item { type int32; }, it’s expected that <item>+3</item> in XML gets converted to "item": 3 in JSON, and by that the value representation loses its original non-canonical string representation.

For leaf item { type string; }, it’s expected that <item>+3</item> in XML gets converted to "item": "+3" in JSON.

Transcoders MAY choose to warn users when they are converting a non-canonical value representation.

TODO: add CBOR examples

3.1. The union Built-In Type

According to Section 9.12 of [RFC7950], it’s possible to specify that a value may be of several types. Transcoders SHOULD perform a conceptual round-trip of converting such value to the XML representation and back, to check that the resulting value is indeed interpreted as the same type as in the original representation.

Example:

```
leaf item {
  type union {
    type int32;
    type string;
  }
}
```

If the received JSON data is "item": "+3", the type is to be interpreted as a string, according to Section 6.10 of [RFC7951]. Yet, after encoding to XML, the type information is lost and the value gets reinterpreted as a number.

Transcoders SHOULD issue a warning every time a union-type value is reinterpreted as another member type, to avoid possible user confusion, unless the Original Type Annotation is used and known to be understood by the other party.

3.2. Netconf Reinterpretation Warning

An additional error tag is specified for NETCONF, and by extension to RESTCONF and CORECONF.

```
error-tag:      reinterpreted-value
error-type:     transport, rpc, protocol, application
error-severity: warning
error-info:     <bad-element> : identifies the elements in the data model
                  which the proxy had to reinterpret to another type.
                  May appear multiple times.
Description:    An element value has been supplied with an additional type information
                which could not be conveyed to the final destination, and by dropping
                this information, the value got reinterpreted as a different type.
```

TODO: properly augment ietf-netconf:error-tag-type.

3.3. Original type annotation

In order to keep the type information with the value even through conversion, transcoders (and generally encoders) MAY attach a type annotation to such a value. The following module defines the "original-type" annotation.

```
module ietf-yang-original-type {
  prefix "ot";
  import ietf-yang-metadata {
    prefix "md";
  }
  md:annotation original-type {
    type string;
    description "This annotation contains the original type of the annotated value.";
  }
}
```

Example: <item ot:original-type="string">+3</item>

If the Original Type Annotation is used and understood, contrary to what is specified in Section 9.12 of [RFC7950], the received value MUST be validated first against the specified original type, and SHOULD NOT be accepted as any other member type, even if a match is found.

By the same logic, this extends Section 6.10 of [RFC7951], so that not only the implicit encoding of the value is taken into account, but also the Original Type Annotation if present.

TODO: make an annotation draft for CBOR by tagging keys.

When encoding and decoding in and from CBOR with Standins ([I-D.bormann-cbor-yang-standin]), a standin SHOULD be used whenever possible.

Specifying the Original Type Annotation also resolves possible problems with unions containing leafref as a member type. The example in Section 9.12.4 of [RFC7950] becomes much simpler by explicitly indicating whether the value is intended to be interpreted as a leafref or something else. Most notably, the original specification requires revalidation every time the target instance may have been added or deleted, which may be error-prone for development and costly for operations.

4. Proxying NETCONF, RESTCONF and CORECONF protocols

If the node behaves as a proxy between different protocols, it should translate as wide range of requests as realistically possible.

When the request is impossible to be fulfilled because the underlying protocol lacks the needed capabilities, the proxy MUST indicate this failure by a rpc-error response with the error-tag set to operation-not-supported.

For example, the NETCONF operations <lock> and <unlock> are not defined for RESTCONF, and therefore the implementation needs to decide whether to plainly refuse them, or to somehow emulate the overall effect.

When a single request can’t be translated to a single underlying request, the proxy may instead perform multiple consecutive requests. If any of these requests fails, the proxy MUST issue appropriate rollback requests, and properly relay all errors, so that the original request’s semantics is kept intact.

For example, if a complex RESTCONF PATCH operation can’t be translated to one NETCONF request, the proxy may choose to issue <lock>, perform multiple edits and finalize by <unlock>. If such an edit fails half-way, the proxy must revert it before unlocking and failing, so that the RESTCONF operation fails cleanly.

Detailed specification of request and response translation is out of scope of this document, and may depend on exact context and even be specific for the YANG models supported.

5. Operational Considerations

Running proxies always adds inherent complexity into the network, and it may be better to look for tools natively supporting the required protocol, instead of deploying a proxy.

For device vendors, running a proxy may be an only solution. If a device exposes a summary endpoint by a proxy over several sub-endpoints, it is recommended to not expose the sub-endpoints even just for debug purposes, as accessing a sub-endpoint directly may break some consistency expectations on the proxy.

6. Security Considerations

It’s required by Section 2.2 of [RFC6241] that all NETCONF connections provide authentication, data integrity, confidentiality and replay protection. While different proxies may handle these aspects differently, it MUST be documented how these are handled.

Specifically if authentication is terminated at a proxy, the underlying endpoints MUST be isolated in such a way that they are only accessible through the proxy.

7. IANA Considerations

TODO: Do we need to require registration of the YANG module for the Original Type Annotation?

Otherwise: This document has no IANA actions.

8. Normative References

[I-D.bormann-cbor-yang-standin]
Bormann, C. and M. Matト魎ka, "Stand-in Tags for YANG-CBOR", Work in Progress, Internet-Draft, draft-bormann-cbor-yang-standin-02, 30 August 2025, <https://datatracker.ietf.org/doc/html/draft-bormann-cbor-yang-standin-02>.

[I-D.ietf-core-comi]
Veillette, M., Van der Stok, P., Pelov, A., Bierman, A., and C. Bormann, "CoAP Management Interface (CORECONF)", Work in Progress, Internet-Draft, draft-ietf-core-comi-20, 6 May 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-core-comi-20>.

[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>.

[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>.

[RFC7951] Lhotka, L., "JSON Encoding of Data Modeled with YANG", RFC 7951, DOI 10.17487/RFC7951, August 2016, <https://www.rfc-editor.org/rfc/rfc7951>.

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

[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>.

[RFC9254] Veillette, M., Ed., Petrov, I., Ed., Pelov, A., Bormann, C., and M. Richardson, "Encoding of Data Modeled with YANG in the Concise Binary Object Representation (CBOR)", RFC 9254, DOI 10.17487/RFC9254, July 2022, <https://www.rfc-editor.org/rfc/rfc9254>.

Acknowledgments

This document is a result of prior work of Maria Matト魍ka and Carsten Bormann on [I-D.bormann-cbor-yang-standin], and design discussions with Vojtト嫩h Vilテュmek over a NETCONF-RESTCONF-CORECONF proxy.

Author’s Address

Maria Matejka
CZ.NIC
Milesovska 1136/5
13000 Praha
Czechia
Email: maria.matejka@nic.cz, mq@jmq.cz