

OD-360 – Broadband Forum YANG Modules

Issue: 1 Amendment 1

Issue Date: March 2023

- [References](#)
- [Overview](#)
 - [YANG Language Version](#)
- [YANG Guidelines](#)
 - [Terminology](#)
 - [Template](#)
 - [IETF YANG Guidelines](#)
 - [IETF-1: YANG Terms](#)
 - [IETF-2: YANG Validation](#)
 - [IETF-3: YANG Usage Guidelines](#)
 - [IETF-4: File Layout](#)
 - [IETF-5: Quoting](#)
 - [IETF-6: Module Naming Conventions](#)
 - [IETF-7: Module Header, Meta and Revision Statements](#)
 - [IETF-8: Module Header, Meta and Revision Statements - Top Level Description](#)
 - [IETF-9: Namespace Assignments](#)
 - [IETF-10: Prefixes - Imported Modules](#)
 - [IETF-11: Prefixes - Usage](#)
 - [IETF-12: Identifier Naming Conventions](#)
 - [IETF-13: Conditional Statements](#)
 - [IETF-14: Conditional Augment Statements](#)
 - [IETF-15: Import or Include By Revision](#)
 - [IETF-16: Legal Characters in YANG Modules](#)
 - [IETF-17: Prefix on “belongs-to” Statements](#)
 - [IETF-18: ‘error-message’ sub-statement to the ‘must’ statement](#)
 - [IETF-19: ‘description’ sub-statement to the ‘must’ statement](#)

- [IETF-20: Tabs and Spacing](#)
- [BBF YANG Guidelines](#)
 - [BBF-1: Use of IANA/IETF YANG Modules](#)
 - [BBF-2: Line Length](#)
 - [BBF-3: Deviations](#)
 - [BBF-4: Short Names](#)
 - [BBF-5: Descriptions and References](#)
 - [BBF-6: Special Values](#)
 - [BBF-7: Paragraph Separation in Description Statements](#)
 - [BBF-8: Revision Statements](#)
 - [BBF-9: Explicit Modeling](#)
 - [BBF-10: Retroactive Application of OD-360 Guidelines](#)
 - [BBF-11: Multi-word Identifiers](#)
 - [BBF-12: Acronyms in Description Statements](#)
 - [BBF-13: Abbreviations in Description Statements](#)
 - [BBF-14: Adding and/or Creating BBF YANG Models](#)
 - [BBF-15: YANG Model Revisions and License](#)
 - [BBF-16: Enum Naming](#)
 - [BBF-17: Value Statements](#)
 - [BBF-18: Prefix on 'if-feature' Statements](#)
 - [BBF-19: Default Case of a Choice Statement - Empty Typed Leaf](#)
 - [BBF-20: Formatting 'error-string' statements](#)
- [BBF Deviation Guidelines](#)
 - [Types of Deviations](#)
 - [DEV-1: not-supported](#)
 - [DEV-2: add - Number of Elements](#)
 - [DEV-3: add - 'must' Constraints](#)
 - [DEV-4: replace - Data Type](#)
 - [DEV-5: replace - Number of Elements](#)
 - [DEV-6: delete](#)
- [BBF Guidelines for 'units' Statements](#)
 - [Terminology](#)
 - [UNITS-1: Unit vs Symbol](#)
 - [UNITS-2: Plural Form](#)
 - [UNITS-3: Base vs Derived](#)
 - [UNITS-4: Memory](#)
 - [UNITS-5: Unit Combinations](#)

- [UNITS-6: Equivalent Units](#)
- [UNITS-7: Use of decimal64](#)
- [UNITS-8: Exceptions](#)
- [UNITS-9: Frames vs Packets vs Messages](#)

List of Figures

List of Tables

Notice

The Broadband Forum is a non-profit corporation organized to create guidelines for broadband network system development and deployment. These Guidelines have been approved by members of the Forum. These Guidelines are subject to change. These Guidelines are owned and copyrighted by the Broadband Forum, and all rights are reserved. Portions of these Guidelines may be owned and/or copyrighted by Broadband Forum members.

Intellectual Property

Recipients of these Guidelines are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of these Guidelines, or use of any software code normatively referenced in these Guidelines, and to provide supporting documentation.

Terms of Use

1. License

Broadband Forum hereby grants you the right, without charge, on a perpetual, non-exclusive and worldwide basis, to utilize the Guidelines for the purpose of developing, making, having made, using, marketing, importing, offering to sell or license, and selling or licensing, and to otherwise distribute, products complying with the Guidelines, in all cases subject to the conditions set forth in this notice and any relevant patent and other intellectual property rights of third parties (which may include members of Broadband Forum). This license grant does not include the right to sublicense, modify or create derivative works based upon the Guidelines except to the extent these Guidelines include text implementable in computer code, in which case your right under this License to create and modify derivative works is limited to modifying and creating derivative works of such code. For the avoidance of doubt, except as qualified by the preceding sentence, products implementing these Guidelines are not deemed to be derivative works of the Guidelines.

2. NO WARRANTIES

THESE GUIDELINES ARE BEING OFFERED WITHOUT ANY WARRANTY WHATSOEVER, AND IN PARTICULAR, ANY WARRANTY OF NONINFRINGEMENT AND ANY IMPLIED WARRANTIES ARE EXPRESSLY DISCLAIMED. ANY USE OF THESE GUIDELINES SHALL

BE MADE ENTIRELY AT THE USER'S OR IMPLEMENTER'S OWN RISK, AND NEITHER THE BROADBAND FORUM, NOR ANY OF ITS MEMBERS OR SUBMITTERS, SHALL HAVE ANY LIABILITY WHATSOEVER TO ANY USER, IMPLEMENTER, OR THIRD PARTY FOR ANY DAMAGES OF ANY NATURE WHATSOEVER, DIRECTLY OR INDIRECTLY, ARISING FROM THE USE OF THESE GUIDELINES, INCLUDING BUT NOT LIMITED TO, ANY CONSEQUENTIAL, SPECIAL, PUNITIVE, INCIDENTAL, AND INDIRECT DAMAGES.

3. THIRD PARTY RIGHTS

Without limiting the generality of Section 2 above, BROADBAND FORUM ASSUMES NO RESPONSIBILITY TO COMPILE, CONFIRM, UPDATE OR MAKE PUBLIC ANY THIRD PARTY ASSERTIONS OF PATENT OR OTHER INTELLECTUAL PROPERTY RIGHTS THAT MIGHT NOW OR IN THE FUTURE BE INFRINGED BY AN IMPLEMENTATION OF THE GUIDELINES IN ITS CURRENT, OR IN ANY FUTURE FORM. IF ANY SUCH RIGHTS ARE DESCRIBED ON THE GUIDELINES, BROADBAND FORUM TAKES NO POSITION AS TO THE VALIDITY OR INVALIDITY OF SUCH ASSERTIONS, OR THAT ALL SUCH ASSERTIONS THAT HAVE OR MAY BE MADE ARE SO LISTED.

All copies of these Guidelines (or any portion hereof) must include the notices, legends, and other provisions set forth on this page.

Issue History

Issue Number	Approval Date	Issue Editor	Changes
Issue 1	15 February 2022	Joey Boyd, Adtran	Original
Issue 1 Amendment 1	15 March 2023	Joey Boyd, Adtran	Add 'units' statement guidelines

Comments or questions about this Broadband Forum document should be directed to info@broadband-forum.org.

References

- [1] [RFC 7223](#), *A YANG Data Model for Interface Management*, IETF, 2014
- [2] [RFC 7950](#), *The YANG 1.1 Data Modeling Language*, IETF, 2016
- [3] [RFC 8407](#), *Guidelines for Authors and Reviewers of Documents Containing YANG Data Models*, IETF, 2018

Overview

OD-360 provides a set of Best Current Practices for development of YANG Data Models in the Broadband Forum (BBF). The approach taken in establishing a set of best current practices is to utilize existing practices where those are relevant to the work of the Forum.

YANG Language Version

The BBF has adopted YANG 1.1 [2] for use in all BBF YANG Data Models.

YANG Guidelines

These YANG guidelines are organized as follows:

- Those based on IETF YANG Guidelines
- Additional BBF-specific guidelines

Terminology

Throughout these guidelines, the term module can be used as a generic term for a YANG module or submodule. When describing properties that are specific to submodules, the term submodule is used instead.

Template

New YANG modules MAY be created from the BBF template as shown here [bbf-template.yang](#). This template is based on the IETF YANG module template defined in the IETF YANG Guidelines.

IETF YANG Guidelines

The general principle is that the IETF YANG Guidelines, defined in RFC 8407[3], apply in their entirety to BBF YANG modules. However, they are aimed at IETF Standards Track YANG modules and contain some IETF specific guidelines that cannot be applied literally to BBF YANG modules.

- Sections 1 (introduction), 2 (Terminology) and 3 (General Documentation Guidelines) are mostly IETF-specific and have little impact on YANG module definitions. These sections do not apply unless specifically noted.
- Section 4 (YANG Usage Guidelines) is mostly IETF-independent and has major impact on YANG module definitions. This section applies unless specifically noted.
- Sections 5 (IANA Considerations), 6 (Security Considerations), 7 (Acknowledgements), 8 (References) and Appendices A (Module Review Checklist) and B (YANG Module Template) are mostly IETF-specific and have little impact on YANG module definitions. These sections do not apply unless specifically noted.

IETF-1: YANG Terms

This requirement extends section 2.2 of RFC 8407 [3].

BBF YANG modules MUST use YANG terminology, e.g., when used in descriptions the terms SHOULD refer to “data nodes”, “containers”, “lists”, etc. For example, do not use the term “objects” and instead refer to “data nodes”.

IETF-2: YANG Validation

This requirement overrides section 3.10 of RFC 8407 [3].

BBF YANG modules MUST be validated using pyang 2.3.2 (or later). Validation MUST use the various “lint” options described below.

```
--lint
--lint-modulename-prefix=bbf
--lint-namespace-prefix=urn:bbf:yang:
--max-line-len=70
```

pyang can be installed via PyPi.

The GitHub version of the pyang tool is available at <https://github.com/mbj4668/pyang>

There is a docker image available that contains an installation of pyang, yanglint and yangerc.
This image may, at times, include a newer version of pyang than is stated in this guideline.
<https://hub.docker.com/r/broadbandforum/yangtools>

IETF-3: YANG Usage Guidelines

This requirement restates section 4 of RFC 8407 [3] in the context of the BBF.

BBF YANG modules comply with all syntactic and semantic requirements of YANG 1.1 [2].

IETF-4: File Layout

This requirement overrides section 5.2 of RFC 7950 [2].

Each BBF YANG module or submodule MUST be stored in a file called <module-name>.yang or <submodule-name>.yang. BBF YANG modules and submodules will not be stored with the revision-date in the name.

IETF-5: Quoting

This requirement is an extension to section 6.1.3 of RFC 7950 [2].

The rules specified in this section effectively mean that the second and subsequent lines of multi-line strings have to be aligned under the character following the opening double quote.

```
leaf multi-line-example {  
    type string;  
    description  
        "This is the first line of a multi-line string.  
        This is the alignment of the second and subsequent lines."  
}
```

IETF-6: Module Naming Conventions

This requirement overrides section 4.1 of RFC 8407 [3].

<module-name> or <submodule-name> is the module name or submodule name, which (see below) MUST begin with the prefix “bbf”.

The remainder of the section applies with appropriate translations of IETF terms to BBF terms, e.g., “IETF” and “IANA” are read as “BBF”, and the “ietf-” prefix is read as “bbf-”.

IETF-7: Module Header, Meta and Revision Statements

These requirements override section 4.8 of RFC 8407 [3].

namespace

See IETF-9.

organization

MUST be of the form:

```
Broadband Forum <https://www.broadband-forum.org>  
<work area name> Work Area";
```

or

```
Broadband Forum <https://www.broadband-forum.org>  
<project stream name> Project Stream";
```

contact

MUST use the following text:

```
Comments or questions about this Broadband Forum YANG module  
should be directed to <mailto:help@broadband-forum.org>.
```

Followed by:

- Editor name(s) and affiliation(s)
- Project Stream Leader name(s) and affiliation(s) - omit if no Project Stream Leader
- Work Area Director name(s) and affiliation(s)

Editor: <name>, <company>

Editor: <name>, <company>

PS Leader: <name>, <company>

PS Leader: <name>, <company>

WA Director: <name>, <company>

WA Director: <name>, <company>

description

See IETF-8.

revision

The requirements from RFC 8407 apply with the following modifications:

- The requirements relating to the reference substatement are replaced with a requirement to reference the associated TR Issue, Amendment(if any) and Corrigendum.
- The requirements relating to reuse of the same revision statement within unpublished versions are reworded to refer only to “versions” and not to “Internet-Drafts”.

References to associated TRs use “abbreviated” TR names, e.g., “TR-101i2” or “TR-383a6”.

IETF-8: Module Header, Meta and Revision Statements - Top Level Description

This requirement overrides section [4.8](#) of RFC 8407 [\[3\]](#).

The top-level description MUST contain the following:

- Summary
- BBF Notice Section

IETF-9: Namespace Assignments

This requirement overrides section [4.9](#) of RFC 8407 [\[3\]](#).

Section 4.9 applies with appropriate translations of IETF terms to BBF terms, e.g., “IANA” is read as “BBF” and “non-Standards-Track” is read as “example” or “not-for-publication”.

BBF YANG module namespace MUST be of the form:

```
urn:bbf:yang:<module-name>
```

IETF-10: Prefixes - Imported Modules

This requirement extends section [4.2](#) of RFC 8407 [\[3\]](#).

Imported modules MUST have the same prefix defined in the module they are imported into as is defined in the imported module.

Example:

```
module bbf-example1 {
  namespace "urn:bbf:yang:bbf-example1";
  prefix bbf-ex1;
  ...

  module bbf-example2 {
    namespace "urn:bbf:yang:bbf-example2";
    prefix bbf-ex2;

    import bbf-example1 {
      prefix bbf-ex1; //same prefix as defined in the module, bbf-
example1.
    }
    ...
```

IETF-11: Prefixes - Usage

These requirements restate section [4.2](#) of RFC 8407 [\[3\]](#).

The following apply to prefix usage of the local module:

- The local module prefix SHOULD be used instead of no prefix in all path expressions. The exceptions to this are path statements defined in a grouping that reference a node in another grouping. The referenced node will belong to the module(s) where the grouping is used, so the prefix cannot be assumed.
- The local module prefix MUST be used instead of no prefix in all default statements for an identityref or instance-identifier data type.
- The local module prefix MAY be used for references to typedefs, groupings, extensions, features and identities defined in the module.

IETF-12: Identifier Naming Conventions

The following extends section 4.3.1 of RFC 8407 [3].

These guidelines are stronger than the IETF guidelines because they do not allow upper-case characters and the underscore character:

- Only lower-case letters, numbers, dashes “-” and dots “.” MUST be used in identifier names.
- Identifiers SHOULD include complete words and/or well-known acronyms or abbreviations. There are some exceptions to this for performance reasons. See BBF-5 for further guidelines.

IETF-13: Conditional Statements

The following reiterates section 4.5 of RFC 8407 [3].

As stated, conditional requirements for data nodes must be documented somewhere. Where possible, use YANG modeled constraints such as an ‘if-feature’ to make nodes explicitly optional or a ‘when’ statement to constrain data based on other data nodes. At a minimum, the description should be used to convey a constraint which cannot be modeled.

IETF-14: Conditional Augment Statements

The following reiterates section 4.19.1 of RFC 8407 [3].

Section 4.19.1 states that the augment statement is often used together with the ‘when’ statement and/or ‘if-feature’ statement to make the augmentation conditional on some portion of the data model.

Where possible, all BBF YANG models SHOULD apply ‘when’ and/or ‘if-feature’ statements to each augment.

IETF-15: Import or Include By Revision

The following replaces statements in section 4.7 of RFC 8407 [3] regarding the use of the ‘revision-date’ substatement.

BBF YANG modules MUST NOT use ‘revision-date’ on import and include statements.

IETF-16: Legal Characters in YANG Modules

The following replaces portions of section 6 of RFC 7950 [2].

BBF YANG modules MUST be written using only printable ASCII characters.

IETF-17: Prefix on “belongs-to” Statements

The following extends section 4.2 of RFC 8407 [3].

In a submodule, the prefix defined for the parent module in the ‘belongs-to’ statement MUST match the prefix defined in the parent module for itself.

```
module bbf-example1 {  
    namespace "urn:bbf:yang:bbf-example1";  
    prefix bbf-ex1;  
  
    include bbf-example1-sub1;  
    ...  
}
```

```
submodule bbf-example1-sub1 {  
  belongs-to bbf-example1;  
  prefix bbf-ex1;  
}  
...
```

IETF-18: 'error-message' sub-statement to the 'must' statement

This requirement overrides section [7.5.4.1](#) of RFC 7950 [2].

The 'error-message' sub-statement MUST be provided to 'must' statements in configuration data nodes for proper communication of the constraint validation failure.

IETF-19: 'description' sub-statement to the 'must' statement

This requirement overrides section [7.5.4](#) of RFC 7950 [2].

The 'description' sub-statement MUST be provided to the 'must' statement for proper communication of the intent of the statement.

IETF-20: Tabs and Spacing

This requirement is an extension to section [6.1.3](#) of RFC 7950 [2].

BBF YANG modules MUST NOT contain any tab characters and MUST use 2 space characters for indentation.

BBF YANG Guidelines

This section adds BBF-specific guidelines that go beyond the scope of the IETF YANG Guidelines.

BBF-1: Use of IANA/IETF YANG Modules

BBF YANG modules **MUST** use standard IANA/IETF YANG modules whenever possible. In this context, “use” implies adherence to the letter and spirit of such modules and of their defining RFCs.

Individual Working Text documents **SHOULD** reference the standard models which are applicable.

BBF-2: Line Length

The length of each line of text in a YANG module **MUST** not exceed 70 characters.

BBF-3: Deviations

Deviations **MUST NOT** be used in BBF YANG modules.

However, deviations can be used alongside BBF YANG modules as described in the section BBF Deviation Guidelines.

BBF-4: Short Names

Lists and leaf-lists may have many elements resulting in large amounts of data present on the line. For XML encoding, the name of each node appears twice for every element of the list or leaf-list. Although IETF-12 states that identifiers **SHOULD** include complete words, the expedient use of short names **SHOULD** be imposed to shorten on-the-wire messaging and improve efficiency.

BBF-5: Descriptions and References

All descriptions **MUST** read as sentences or sentence fragments containing proper capitalization and punctuation, e.g., ending with periods. However, reference statements need not meet this requirement and do not need to terminate with a period.

BBF-6: Special Values

Special values **SHOULD** be parameterized in an enumeration as part of a union with the normal

values.

Enumerations SHOULD NOT be used in cases where a special value means “XXX or less” or “YYY or more”, i.e. where the special value indicates clipping.

Example:

```
typedef snr-margin {
    type union {
        type enumeration {
            enum "undetermined" {
                description
                    "Indicates the value is not determined.";
            }
        }
        type int16 {
            range "-511..511";
        }
    }
    description
        "Reports the signal-to-noise ratio margin. A first special value
        (undetermined) indicates that the signal-to-noise ratio margin
        is undetermined. A second special value (-511) indicates that
        the signal-to-noise ratio margin is less than or equal to -51.1
        dB. A third special value (+511) indicates that the
        signal-to-noise ratio margin is greater than or equal to
        +51.1dB.";
```

In the above example, -511 refers to a value less than or equal to -51.1; +511 refers to a value greater than or equal to +51.1 and the enum undetermined refers to a value which is undetermined.

BBF-7: Paragraph Separation in Description Statements

For description statements which contain multiple paragraphs, each paragraph SHOULD be separated using a blank line.

Example:

description

"This is the first paragraph of the description.

This is the second paragraph of the description.";

BBF-8: Revision Statements

Unpublished versions of BBF YANG modules MAY have 'revision' statements corresponding to individual pull requests. However, published versions MUST remove these 'revision' statements and replace with a single 'revision' statement corresponding to the published version.

BBF-9: Explicit Modeling

Where YANG allows something or some behavior to be explicitly modeled (e.g., enumerations, range, units, must, etc.) then this SHOULD always be done. Mandatory requirements in descriptions SHOULD be used only where it is not possible to use formal YANG modeling to convey the requirement.

BBF-10: Retroactive Application of OD-360 Guidelines

OD-360 guidelines SHOULD NOT be retroactively applied to published BBF YANG modules and MUST NOT be applied if the resulting change is backward incompatible with the previously published version. For example, an existing node named using an underscore '_' instead of a dash '-' MUST NOT be changed in published module as the resulting renaming is not backward compatible.

BBF-11: Multi-word Identifiers

YANG identifiers which represent multiple words, e.g., target margin, SHOULD use a dash between the normally whitespace separated words, e.g., 'target-margin'.

BBF-12: Acronyms in Description Statements

Acronyms can be used in description statements. However, the first instance of the acronym in a

description statement SHOULD have its meaning clearly conveyed.

For example:

```
leaf cbs {  
  type bbf-qos-plc-tp:burst-size;  
  description  
    "Committed Burst Size (CBS) defines the amount of traffic  
    that can be admitted above the Committed Information Rate  
    (CIR) and considered green."  
}
```

BBF-13: Abbreviations in Description Statements

Abbreviations can be used in description statements. However, the first instance of the acronym in a description statement SHOULD have its meaning clearly conveyed.

BBF-14: Adding and/or Creating BBF YANG Models

This guideline defines the best practices when defining a new BBF YANG model and/or having BBF extend an existing (BBF or non-BBF) YANG model.

When creating or editing BBF YANG models, the following use cases apply:

1. **Requirements that are captured in a Broadband Forum Technical Report** In this case, the BBF YANG model should meet the requirement set forward in the Technical Report. For example, the models contained in TR-383 build on the requirements of TR-101/156/167/301.
2. **Requirements (or information models) that are captured in standards documents published by other SDOs and referenced from a Broadband Forum Technical Report, for which the other SDO is not developing a YANG model:** In this case, the BBF YANG data model should reference the applicable standard and be able to meet the requirement(s) from that document. For example, the TR-355 G.fast YANG model contains attributes defined in ITU-T Recommendation G.997.2.
3. **A model requirement which is not formulated in a Broadband Forum Technical Report or a standard published by another SDO:** In this case, the following guidelines apply:

1. First, the requirement must be formulated based on inputs brought forward to the BBF (i.e. a contribution explaining the need for the requirement).
2. There must be consensus among the participants in the Work Area/Project Stream to move forward with this requirement.
3. It is desirable, but not mandatory, for the requirements to be captured in a BBF Technical Report. However, that should not hold up putting the requirement in the YANG model. In other words, if the Work Area agrees that something should be added in a BBF YANG model, then we do not need to wait until the requirement is captured in a Technical Report.
4. The new requirement must not break existing BBF Technical Reports.
5. The new requirement must be reviewed by any Work Area(s) or Project Stream(s) for which it is clearly applicable (e.g., a PON related requirement is to be reviewed by the FAN WA).

BBF-15: YANG Model Revisions and License

BBF YANG Models are available in three possible states:

- Published: Models are made available to the public in GitHub and are associated with a Technical Report.
- Draft: Models are made available to the public in GitHub and are associated with a draft Working Text.
- Development: Models are made available to members in Bitbucket while they are in development.

In order to allow BBF members to consume BBF YANG models in any state, the following apply with regards to revision statements and license text.

Published

Each published module and submodule SHALL:

- have a new, if previously published, or initial revision statement containing the TR publication date as the revision date. All submodules associated with a module SHALL have the same revision date as the parent module regardless of whether they were changed.
- maintain all revision statements from previously published revisions.

- contain the standard BBF software license.

Draft

Each draft module and submodule SHALL:

- have a new, if previously published, or initial revision statement containing the draft publication date as the revision date. All submodules associated with a module SHALL have the same revision date as the parent module regardless of whether they were changed.
- maintain all revision statements from previously published revisions.
- contain the draft BBF software license.

Development

Each development module and submodule SHALL:

- have a new, if previously published, or initial revision statement containing the date of the end of the review period of the last pull request that contained the module and/or submodule. All submodules associated with a module SHALL have the same revision date as the parent module regardless of whether they were changed as part of the recent pull request.
- maintain all revision statements from previously published revisions.
- contain the standard BBF software license with correct copyright year(s).

The end of the license must state the following (replacing xxx with the Working Text number):

```
This version of this YANG module is part of WT-xxx; see
the WT itself for full legal notices.";
```

The format of the revision statement in a development module shall be as follows:

```
revision 2021-09-03 {
  description
    "TBD
    * Approval Date:    TBD.
    * Publication Date: TBD.";
  reference
    "TBD";
}
```

BBF-16: Enum Naming

All enums in an enumeration SHOULD be named using names that describe their functional meaning rather than something ambiguous such as an integer string, "1".

For example, this definition is preferred

```
leaf ra-mode {  
  type enumeration {  
    enum manual {  
      description  
        "Mode 1 = MANUAL.";  
    }  
    enum at-init {  
      description  
        "Mode 2 = AT_INIT.";  
    }  
    enum dynamic {  
      description  
        "Mode 3 = DYNAMIC.";  
    }  
    enum dynamic-with-sos {  
      description  
        "Mode 4 = DYNAMIC with SOS.";  
    }  
  }  
}
```

over this one.

```
leaf ra-mode {  
  type enumeration {  
    enum 1 {  
      description  
        "Mode 1 = MANUAL.";  
    }  
    enum 2 {  
      description  
        "Mode 2 = AT_INIT.";  
    }  
    enum 3 {  
      description  
        "Mode 3 = DYNAMIC.";  
    }  
    enum 4 {  
      description  
        "Mode 4 = DYNAMIC with SOS.";  
    }  
  }  
}
```

BBF-17: Value Statements

When modeling an enumeration where each enum corresponds to an integer value specified by a underlying standard specification, a model MAY choose to model those values in the 'value' statement.

The 'value' carries no meaning via NETCONF nor RESTCONF but is used for an implementation and reference aid.

For an enumeration whose underlying values are 0..n, the 'value' statements are not necessary as all enums have an implied value if not explicitly specified. The implied values begin with 0 and increment by 1 for each enum in the enumeration.

```
leaf ra-mode {  
  type enumeration {  
    enum manual {  
      value 1;  
      description  
        "Mode 1 = MANUAL.";  
    }  
    enum at-init {  
      value 2;  
      description  
        "Mode 2 = AT_INIT.";  
    }  
    enum dynamic {  
      value 3;  
      description  
        "Mode 3 = DYNAMIC.";  
    }  
    enum dynamic-with-sos {  
      value 4;  
      description  
        "Mode 4 = DYNAMIC with SOS.";  
    }  
  }  
}
```

BBF-18: Prefix on 'if-feature' Statements

An 'if-feature' statement on any node defined within a top level grouping which depends on a 'feature' that is defined in the same module as the grouping MUST use the prefix on the feature name contained within the 'if-feature' statement.


```
module bbf-availability {  
    ...  
  
    feature availability {  
        description  
            "Indicates support for retrieving availability of resources.";  
    }  
  
    ...  
  
    grouping availability {  
        description  
            "Defines the availability of entities.";  
  
        container availability {  
            if-feature "bbf-avail:availability";  
            presence  
                "If present, this container indicates supports for  
                retrieving the availability of the entity for which it is  
                used.";  
            config false;  
            description  
                "Operational status defining the availability of an  
                entity.";  
  
            uses availability-parameters;  
        }  
    }  
  
    ...  
}
```

BBF-19: Default Case of a Choice Statement - Empty Typed Leaf

This requirement is derived from section [7.9.3](#) and section [9.11](#) of RFC 7950 [2].

The default case of a choice statement **MUST** not contain a sole descendant leaf node of type “empty”.

Section 7.9.3 states that “The default case is only important when considering the ‘default’ statements of nodes under the cases (i.e., default values of leafs and leaf-lists, and default cases of nested choices).” It also states that “The default values and nested default cases under the default case are used if none of the nodes under any of the cases are present.”.

Section 9.11 states that an empty type cannot have a default value

In the example below, the choice ‘manual’ has a leaf with an empty type as the only child node. Since an empty leaf cannot have a default value, the behavior of an <edit-config> request that creates the ‘transfer’ container without providing any data for the choice ‘how’ will be the same irrespective of whether the default ‘manual’ statement is present or not.

The leaf ‘manual’ will not be created.

```
container transfer {
  description
    "Configuration associated with the timing
    of a transfer.";

  choice how {
    default manual;
    description
      "Transfer timing methods.";

    case interval {
      leaf interval {
        type uint16;
        units minutes;
        default 30;
        description
          "Transfer every 'interval' minutes.";
      }
    }

    case manual {
      leaf manual {
        type empty;
        description
          "Only transfer when manually triggered.";
      }
    }
  }
}
```

BBF-20: Formatting 'error-string' statements

To ensure ease of rendering in client tools such as GUIs etc. and regardless of their length, error-messages MUST be defined as a single line of text and MUST NOT contain explicit line breaks (\n) or tab (\t) characters or formatted in any way.

'error-string' statements which span across multiple lines MUST be defined by concatenating individual quoted-string statements defined per line.

This is necessary to ensure that the strings will be correctly concatenated and white space retained as intended and rendered correctly in client tools.

Example:

```
error-message
  "The name of a multicast interface to host can not
    be the name of a multicast network interface.";
```

Per RFC 7950 [2], section 6.1.3, this 'error-message' would be reported as:

```
<error-message>The name of a multicast interface to host can not\nbe the
name of a multicast network interface.</error-message>
```

Corrected:

```
error-message
  "The name of a multicast interface to host can not "
+ "be the name of a multicast network interface.";
```

Which would be reported as:

```
<error-message>The name of a multicast interface to host can not be the
name of a multicast network interface.</error-message>
```

BBF Deviation Guidelines

It is understood that there are times in which a device will need to support a subset of a standard YANG model whether it be from BBF or any other SDO. YANG provides a means to explicitly model this subset of support via a deviation. While a deviation allows for almost any change in the original definition, it is important that implementations only allow those deviations which are truly a subset of the original and that do not cause the device to violate the underlying specification on which the YANG model is based. The following sections describe the types of deviations and the guidelines by which they are expected to be used in the Broadband Forum.

Types of Deviations

The YANG *deviate* statement is used to define the manner in which the server deviates from a particular YANG module. There are 4 types of deviations defined as arguments to the deviate statement:

- not-supported
 - Indicates the target node is not implemented by the server.
- add
 - Adds properties to the target node so long as they do not already exist.
- replace
 - Replaces properties of the target node so long as they already exist.
- delete
 - Deletes existing properties of the target node.

DEV-1: not-supported

The deviate *not-supported* argument MAY be used to announce lack of support for an optional node.

An optional node is one which does not have a default value nor is made mandatory by use of the mandatory statement. Consider the following object definition which is defined with neither a default value nor a mandatory statement.

```
leaf error-threshold {  
    type uint32;  
    description  
        "The number of errors, when exceeded, will raise an alarm.";  
}
```

Since this node is optional, it is acceptable to announce the object as not supported.

```
deviation error-threshold {  
    description  
        "The optional error threshold is not supported.";  
    deviate not-supported;  
}
```

DEV-2: add - Number of Elements

The deviate *add* argument MAY be used to add a restriction on the number of elements in *a*leaf-list or *list*.

Adding a restriction on the number of elements is accomplished by adding *amin-elements* and/or *max-elements* statement where they were not previously defined. For example, you have *a*leaf-list which does not contain either a *min-elements* or *max-elements* statement and thus does not impose any limit on the number of elements contained in the *leaf-list*.

```
leaf-list port {  
  type string;  
  description  
    "A port associated with a port group."  
}
```

A particular implementation may want to limit the number to 4. In this case, a deviation can be used to add the *max-elements* statement to the definition.

```
deviation port {  
  description  
    "Limit the maximum number of ports to 4."  
  deviate add {  
    max-elements 4;  
  }  
}
```

DEV-3: add - 'must' Constraints

The deviate *add* argument MAY be used to add a constraint on valid data by the use of the *must* statement.

This constraint may be necessary depending on actual hardware in use. For example, the number of physical interfaces of a particular type will be limited by the number present on the hardware. For this, we could add a constraint on the interface list in *ietf-interfaces* [1] to explicitly announce how many interfaces are supported.

```
deviation "/if:interfaces" {  
  description  
    "There are only 8 G.fast interfaces supported.";  
  deviate add {  
    must "count(interface[type='ianaif:gfast'])=8";  
  }  
}
```

DEV-4: replace - Data Type

The deviate *replace* argument MAY be used to replace a type so long as the replacement type uses the same underlying built-in YANG type and the value represented does not fall outside the range of the definition of the type being replaced.

An integer type can be replaced as long as the replacement uses the same built-in YANG type, e.g., uint32 or int64, and that the range specified falls within the range of the original definition. If the original does not define a range, then any range can be specified in the deviation. If the original defines a range of values, the range of the deviation must fall within this defined range. In other words the new minimum value must be greater than or equal to the original, and the new maximum value must be less than or equal to the original. For example, the following defines an object of type uint32 with a range of values 1 to 999999.

```
leaf error-threshold {  
  type uint32 {  
    range "1..999999";  
  }  
  description  
    "The number of errors, when exceeded, will raise an alarm.";  
}
```

A particular implementation may only allow a range from 10 to 99999. Since this range is a subset of the original range definition, we can replace the uint32 type with another uint32 type that has the new range.

```
deviation error-threshold {  
  description  
    "The supported range of values is 10 to 99999.";  
  deviate replace {  
    type uint32 {  
      range "10..99999";  
    }  
  }  
}
```

A string type can be replaced as long as its replacement is also a string and any specified length or pattern is valid per the original definition. If no length is specified in the original, this means that any length restriction may be added. If a length is specified, it means the new length can be specified so long as the new minimum value is greater than or equal to the original, and the new maximum value is less than or equal to the original. If no pattern is specified in the original, any pattern may be added. If a pattern is specified in the original, the new pattern must be equal to or a subset of the original. For example, the following object is defined as a string with no length restriction and a pattern which allows for any alphanumeric character.

```
leaf name {  
  type string {  
    pattern '[A-Za-z0-9]*';  
  }  
  description  
    "The name of a thing.";  
}
```

A particular implementation may require that the maximum length of the string be 255 characters and that upper case characters are not supported. This is a valid deviation since the new string is shorter than the original and the character set is a subset of the original pattern.


```
deviation name {
  description
    "The name may be no more than 255 lowercase letters and/or
    numbers.";
  deviate replace {
    type string {
      length "0..255";
      pattern '[a-z0-9]*';
    }
  }
}
```

DEV-5: replace - Number of Elements

The deviate *replace* argument MAY be used to further restrict the number of elements in *alist* or *leaf-list*.

For *min-elements*, the new value must be greater than or equal to the original. For *max-elements*, the new value must be less than or equal to the original. Suppose we have the following definition of a *leaf-list* of ports which must have at least 1 and no more 10 entries.

```
leaf-list port {
  min-elements 1;
  max-elements 10;
  type string;
  description
    "A port associated with a port group.";
}
```

Now we have an implementation which must have at least 2 ports but no more than 4 ports configured. Since this falls within the constraints of the original definition, it is a valid deviation.

```
deviation port {  
  description  
    "Specify that at least 2 but no more than 4 ports can be  
    configured.";  
  deviate replace {  
    min-elements 2;  
    max-elements 4;  
  }  
}
```

DEV-6: delete

The deviate *delete* argument MUST NOT be used in a deviation.

By deleting a property, the value space of an object is typically being expanded which means the value accepted by the deviated definition is not valid per the original definition and thus, should never be allowed.

BBF Guidelines for ‘units’ Statements

The following guidelines apply to the specification of ‘units’ statements within Broadband Forum YANG modules. These guidelines are based on the following reference documentation as well as practical usage within Broadband Forum YANG modules:

1. IEEE Std. 260.1
2. [NIST Special Publication 811](#)

Terminology

The following terminology is used throughout these guidelines.

Term	Description
------	-------------

family	refers to a group of 'units' statements representing a common functional area including, but not limited to, time, length, temperature, electricity, speed and memory storage
unit	the full text name of a unit, e.g., seconds, milliwatts
symbol	the abbreviated form of expressing the units, e.g., mW, dBm

UNITS-1: Unit vs Symbol

All 'units' statements SHALL use either the "unit" or the "symbol" as defined in IEEE Std. 260.1 including the proper use of uppercase and lowercase. The chosen method used should be applied consistently within a family of units. In general, the "unit" should be used with the "symbol" being used on a case-by-case basis. For example, use "seconds" or some derivation (see UNITS-3) for time but "dBm" for power level as the term "decibel-milliwatts" is not commonly used.

UNITS-2: Plural Form

Where applicable, the full "unit" text SHALL be written in plural form, e.g., seconds vs. second. This differs from the definitions in IEEE Std. 260.1 but aligns with NIST 811 as well as the current practice in industry standard YANG data models.

UNITS-3: Base vs Derived

The base unit for a given family may not always be the one that provides the best granularity. For example, it would not be common to express 20 nanoseconds as 0.00000002 seconds as doing so would make readability difficult. In those situations, a "unit" derived from the base SHOULD be used where the derived unit is a multiple of 10^3 from the base, e.g., seconds, milliseconds, nanoseconds.

UNITS-4: Memory

For 'units' statements related to memory that need derived units for reasons similar to the usage described in UNITS-3, the derived unit SHALL be a multiple of 2^{10} , e.g., bytes, kibibytes, mebibytes.

UNITS-5: Unit Combinations

'units' statements requiring a combination of units SHALL fully spell out the relationship between the units, e.g., "bits per second" rather than "bits/second". Regarding the applicability of UNITS-2, only the first "unit" will be plural in a combination, e.g., "bits" in "bits per second".

UNITS-6: Equivalent Units

If more than one "unit" definition is applicable for a given type of data, then one should be chosen and applied in a consistent manner. For example, use "bytes" instead of "octets".

UNITS-7: Use of decimal64

For a given data node, a base or "unit" along with the decimal64 type SHOULD be used rather than a derived "unit" along with an integer type.

For example, use

```
leaf rx-power {  
  type decimal64 {  
    fraction-digits 4;  
  }  
  units "watts";  
  description  
    "The receive power reported in watts.";  
}
```

rather than

```
leaf rx-power {  
  type uint32;  
  units "0.1 milliwatts";  
  description  
    "The receive power reported in tenths of a milliwatt.";  
}
```

When applying this guideline, along with UNITS-3 and UNITS-4, to already published data models, strict adherence may not always be possible due to backward compatibility issues. In those cases, express the 'units' relative to the closest base or a derived unit, e.g., "0.1 seconds" or "1/256 milliseconds".

UNITS-8: Exceptions

At times, there may be a need to preserve the relationship with an underlying standard that would result in a non-adherence to these guidelines. Such instances should be carefully evaluated on a case-by-case basis to determine if an exception should be made. For example, in TR-385, several data nodes exist that represent data in "125 microsecond PHY frames". Application of UNITS-1 would result in 'units' statements of either "125 microseconds" or "0.125 milliseconds". However, this would result in a disassociation with the underlying standard where the measurement in "PHY frames" is as equally important as is the representation of time.

UNITS-9: Frames vs Packets vs Messages

The use of "frames", "packets" and "messages" SHOULD be applied in a consistent manner based on the following guidelines:

- use "frames" when referring to layer 2, e.g., Ethernet frames, VLANs, etc.
- use "packets" when referring to layer 3, e.g., IP packets
- use "messages" when referring to protocols, e.g., DHCP messages, IGMP messages, etc.