



OD-360

BBF YANG Best Current Practices

Issue: 1 Amendment 3
February 2026

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	February 2022	<ul style="list-style-type: none">Joey Boyd, Adtran	<ul style="list-style-type: none">Initial version
Amendment 1	March 2023	<ul style="list-style-type: none">Joey Boyd, Adtran	<ul style="list-style-type: none">Add 'units' statement guidelines
Amendment 2	January 2024	<ul style="list-style-type: none">Joey Boyd, Adtran	<ul style="list-style-type: none">Add new IETF guidelines IETF-21 and IETF-22.Add new BBF guidelines BBF-21 through BBF-29.Address minor corrections to existing guidelines.
Amendment 3	February 2026	<ul style="list-style-type: none">Joey Boyd, AdtranSven Ooghe, Nokia	<ul style="list-style-type: none">Add new IETF guideline IETF-23.Add new BBF guidelines BBF-30 through BBF-33.Add new deviation guideline DEV-7.Address minor corrections to existing guidelines.

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

Editors

- Joey Boyd, Adtran
- Sven Ooghe, Nokia

Table of Contents

References	6
Overview	6
YANG Language Version	6
YANG Guidelines	6
Terminology	6
Template	6
IETF YANG Guidelines	7
IETF-1: YANG Terms	7
IETF-2: YANG Validation	7
IETF-3: YANG Usage Guidelines	7
IETF-4: File Layout	7
IETF-5: Quoting	8
IETF-6: Module Naming Conventions	8
IETF-7: Module Header, Meta and Revision Statements	8
IETF-8: Module Header, Meta and Revision Statements - Top Level Description	9
IETF-9: Namespace Assignments	9
IETF-10: Prefixes - Imported Modules	9
IETF-11: Prefixes - Usage	10
IETF-12: Identifier Naming Conventions	10
IETF-13: Conditional Statements	10
IETF-14: Conditional Augment Statements	10
IETF-15: Import or Include By Revision	10
IETF-16: Legal Characters in YANG Modules	11
IETF-17: Prefix on "belongs-to" Statements	11
IETF-18: 'error-message' sub-statement to the 'must' statement	11
IETF-19: 'description' sub-statement to the 'must' statement	11
IETF-20: Tabs and Spacing	11
IETF-21: Normative Language	11
IETF-22: 'leafref' path in a 'grouping'	12
IETF-23: Prefix Naming Conventions	13
BBF YANG Guidelines	14
BBF-1: Use of IANA/IETF YANG Modules	14
BBF-2: Line Length	14
BBF-3: Deviations	14
BBF-4: Short Names	14
BBF-5: Descriptions and References	14
BBF-6: Special Values	14
BBF-7: Paragraph Separation in Description Statements	15
BBF-8: Revision Statements	15
BBF-9: Explicit Modeling	15
BBF-10: Retroactive Application of OD-360 Guidelines	15
BBF-11: Multi-word Identifiers	16
BBF-12: Acronyms in Description Statements	16
BBF-13: Abbreviations in Description Statements	16
BBF-14: Creating and/or Extending BBF YANG Models	16
BBF-15: YANG Model Revisions and License	17

BBF-16: Enum Naming	18
BBF-17: Value Statements	18
BBF-18: Prefix on 'if-feature' Statements	19
BBF-19: Default Case of a Choice Statement - Empty Typed Leaf	20
BBF-20: Formatting 'error-string' statements	21
BBF-21: Deprecating and Obsoleting Data Nodes	21
BBF-22: Backwards Compatibility	22
BBF-23: "enable" vs. "enabled"	22
BBF-24: "oper-state" vs. "oper-status"	22
BBF-25: Reference Statements	22
BBF-26: Body Statements	24
BBF-27: 'list' and 'leaf-list' Naming and Descriptions	25
BBF-28: Description on 'when' statements	25
BBF-29: Use of English Contractions	26
BBF-30: Use of Presence statements	26
BBF-31: Formatting XPath expression arguments of YANG statements	27
BBF-32: Naming choices	27
BBF-33: Mapping of alarm type identifiers to ITU-T X.733 and ITU-T X.736 event types	28
BBF Deviation Guidelines	31
Types of Deviations	31
DEV-1: not-supported	31
DEV-2: add - Number of Elements	31
DEV-3: add - 'must' Constraints	32
DEV-4: replace - Data Type	32
DEV-5: replace - Number of Elements	33
DEV-6: delete	34
DEV-7: add - Mandatory leaf	34
BBF Guidelines for 'units' Statements	35
Terminology	35
UNITS-1: Unit vs Symbol	35
UNITS-2: Plural Form	35
UNITS-3: Base vs Derived	35
UNITS-4: Memory	35
UNITS-5: Unit Combinations	35
UNITS-6: Equivalent Units	36
UNITS-7: Use of decimal64	36
UNITS-8: Exceptions	36
UNITS-9: Frames vs Packets vs Messages	36

List of Figures

List of Tables

References

- [1] TR-383 Amendment 8, *Common YANG Modules for Access Networks*, Broadband Forum, 2024
- [2] RFC 2119, *Key words for use in RFCs to Indicate Requirement Levels*, IETF, 1997
- [3] RFC 7223, *A YANG Data Model for Interface Management*, IETF, 2014
- [4] RFC 7950, *The YANG 1.1 Data Modeling Language*, IETF, 2016
- [5] RFC 8174, *Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words*, IETF, 2017
- [6] RFC 8407, *Guidelines for Authors and Reviewers of Documents Containing YANG Data Models*, IETF, 2018
- [7] RFC 8632, *A YANG Data Model for Alarm Management*, IETF, 2019
- [8] X.733, *Information technology - Open Systems Interconnection - Systems Management: Alarm reporting function*, ITU-T, 1992
- [9] X.736, *Information technology - Open Systems Interconnection - Systems Management: Security alarm reporting function*, ITU-T, 1992

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. The guidelines put forth by this document not only provide guidance for authors of BBF YANG models but are also enforced prior to YANG model publication to ensure a high level of quality and consistency across BBF projects.

YANG Language Version

The BBF has adopted YANG 1.1 [4] 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 [6], **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 guideline extends section 2.2 of RFC 8407 [6].

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 guideline overrides section 3.10 of RFC 8407 [6].

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
```

Alternatively, pyang provides the following BBF-specific option that performs the same checks as the options noted above.

```
--bbf
```

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 yanger. 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 guideline restates section 4 of RFC 8407 [6] in the context of the BBF.

BBF YANG modules **MUST** comply with all syntactic and semantic requirements of YANG 1.1 [4].

IETF-4: File Layout

This guideline overrides section 5.2 of RFC 7950 [4].

Each BBF YANG module or submodule MUST be stored within a BBF YANG repository in a file called <module-name>.yang or <submodule-name>.yang without revision-date in the name.

IETF-5: Quoting

This guideline is an extension to section 6.1.3 of RFC 7950 [4].

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 guideline overrides section 4.1 of RFC 8407 [6].

<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 guidelines override section 4.8 of RFC 8407 [6].

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:info@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 guidelines from RFC 8407 apply with the following modifications:

- The guidelines relating to the reference substatement are replaced with a guideline to reference the associated TR Issue, Amendment (if any) and Corrigendum.
- The guidelines 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 guideline overrides section 4.8 of RFC 8407 [6].

The top-level description MUST contain the following:

- Summary
- BBF Notice Section

IETF-9: Namespace Assignments

This guideline overrides section 4.9 of RFC 8407 [6].

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 guideline extends section 4.2 of RFC 8407 [6].

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 guidelines restate section 4.2 of RFC 8407 [6].

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 where the prefix is optional per section 5.1 of RFC 7950 [4].
- 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 [6].

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 [6].

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 [6].

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 [6] 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 [4].

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 [6].

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 guideline overrides section 7.5.4.1 of RFC 7950 [4].

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 guideline overrides section 7.5.4 of RFC 7950 [4].

Not everyone reading a YANG model is an expert in YANG and/or XPath. In order to provide assistance in understanding the intent, all 'must' statements SHALL contain a description statement. This description should convey the intent of the 'must' condition in easy to understand language.

IETF-20: Tabs and Spacing

This guideline is an extension to section 6.1.3 of RFC 7950 [4].

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

IETF-21: Normative Language

This guideline clarifies section 3.6 of RFC 8407 [6].

BBF YANG modules SHALL NOT use normative keywords in their description statements as defined in RFC 2119 [2] and clarified in RFC 8174 [5]. Explicitly, this means avoiding the use of the keywords in all capital letters, e.g., MUST, SHOULD, SHALL, etc.

This does not apply to the license text included as part of each top-level module description.

IETF-22: 'leafref' path in a 'grouping'

The following extends section 4.13 of RFC 8407 [6].

When defined in a 'grouping', a leaf or leaf-list of type 'leafref' MUST NOT specify a path using a relative XPath statement to a node that exists outside the grouping. Referencing nodes outside of a grouping makes assumptions about where within a schema tree the grouping is to be used and thus limits the reusability of the grouping.

The following example violates this guideline because the leaf 'profile-ref' is referencing the leaf 'name' that exists outside of the 'grouping' statement. For this path to be valid, the grouping has to be used in a schema where the data node hierarchy, specified in the 'path', exists.

```
grouping example-grouping {
  description
    "An example grouping.";
  leaf profile-ref {
    type leafref {
      path '../profiles/profile/name';
    }
    description
      "Reference to a profile.";
  }
}
```

The next example demonstrates the use of a relative path to a node within the grouping, which does not violate the guideline.

```
grouping example-grouping {
  description
    "An example grouping.";
  leaf profile-ref {
    type leafref {
      path '../profiles/profile/name';
    }
    description
      "Reference to a profile.";
  }
}

container profiles {
  description
    "Configuration associated with profiles.";

  list profile {
    key name;
    description
      "A profile.";

    leaf name {
      type string;
      description
        "The name of the profile.";
    }
  }
}
```

IETF-23: Prefix Naming Conventions

This guideline extends section 4.2 of RFC 8407 [6].

Prefixes defined in BBF YANG Modules MUST start with "bbf-", e.g., "bbf-yang" in the module bbf-yang-types.

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 **MUST** 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".

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

Superseded by BBF-15.

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 or practical to use formal YANG modeling to convey the requirement.

Example:

```
leaf forwarder {
  type bbf-l2-fwd:forwarder-ref;
  description
    "A reference to a forwarder.

    The following is a constraint that could not be captured in
    YANG: multiple Maintenance Groups referencing the same
    forwarder are allowed, but only if they have a different
    level.

    Note that there is no BBF requirement for multiple levels
    with up-MEPs. Within BBF context multiple levels on a
    forwarder go together with one level using up-MEPs and a
    higher level using MIPs.";
```

BBF-10: Retroactive Application of OD-360 Guidelines

OD-360 guidelines MAY be retroactively applied to published BBF YANG modules but 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 abbreviation in a description statement SHOULD have its meaning clearly conveyed.

BBF-14: Creating and/or Extending 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 as required, or not formulated at all, for the specific use case 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 deviate from any existing requirements in 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 with correct copyright year(s).

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 with correct copyright year(s).

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 {
  yang-version 1.1;
  namespace "urn:bbf:yang:bbf-availability";
  prefix bbf-avail;
  ...

  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 [4].

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 [4], section 6.1.3, this 'error-message' would be reported as either

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

or

```
<error-message>The name of a multicast interface to host can notbe the name of a multicast network interface.</error-message>
```

depending on how the system interprets a new line.

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-21: Deprecating and Obsoleting Data Nodes

There are times when the need arises to migrate away from using previously published data nodes:

- A newer, possibly more efficient, method has been defined.
- Error(s) existing in the current definition that cannot be resolved in a backward compatible way.
- Data nodes were unnecessary or modeled by mistake, and there is no need to maintain or replace them.

The process for this is as follows:

1. Change the status of the applicable data nodes from "current" to "deprecated" by adding a status statement with the value "deprecated".

2. The deprecated data nodes **MUST** remain in this state for at least two years.
3. Beginning with the next publication after at least two years, the data nodes **MAY** be transitioned from "deprecated" to "obsolete".
4. The obsoleted data nodes **MUST** never be deleted from the YANG model.

BBF-22: Backwards Compatibility

An update to an existing YANG module is said to be backward compatible when the clients running with any older versions of the module are able to continue interoperating with a server running the updated version of the module without any issues.

In general, BBF YANG modules **SHOULD NOT** introduce changes that are defined as backwards incompatible per RFC 7950 [4] section 11. Instead, add new definitions and use methods of deprecating and obsoleting previous definitions per RFC 7950 [4] section 7.21.2 and further clarified by BBF-21.

However, experience has shown that in certain cases, additional considerations must be taken into account to decide if an update is acceptable, such as:

- changes that are backwards incompatible according to RFC 7950 [4] section 11 may nevertheless be compatible in conjunction with other changes made at the same time. Therefore, it may be possible to address a specific problem, through backwards incompatible changes.
- RFC 7950 [4] section 11 specifies a bullet list of changes that are considered backward compatible. A change that is not present in this bullet list could also be considered backward compatible if it meets the generic condition of backward compatibility defined in the first paragraph of this OD-360 guideline. Such cases require a case by case decision. For example adding a 'require-instance false' statement to an existing leafref relaxes the constraint, and therefore could be considered backwards compatible, but adding a new similar leaf that contains this statement is usually considered a better choice.

Before agreeing to such changes to a published YANG module, Work Areas must ensure, to the greatest possible extent, that these changes will indeed be backwards compatible and not impact any existing operator configuration or operations or other SDOs or vendors augmentations of these standardized modules.

BBF-23: "enable" vs. "enabled"

Data nodes used to enable/disable some function, and whose identifier is intended to include the term "enable", either solely or as part of a larger name, **SHALL** use the "enabled" form of the word, e.g., 'enabled' or 'statistics-enabled'. This also applies to the names of enums or bits.

BBF-24: "oper-state" vs. "oper-status"

BBF YANG modules **SHALL** use the term 'oper-state' when defining a data node that reports the operational state. This is applicable whether it is the full name or part of the name of the data node, e.g., 'oper-state', 'oper-state-timestamp'.

BBF-25: Reference Statements

The goal of a reference statement is not only to completely and unambiguously guide the user to the documentation that supports the data node(s) being modeled, but also to enable the user to locate data nodes within YANG modules that implement specific attributes or parameters specified in the given documentation by searching through the YANG modules for references to those attributes or parameters. To that end, the following guideline is provided.

Reference statements SHALL reference technical specifications in a consistent manner using one or both of the following formats:

reference

```
"[<single-reference>; <EOL>]* <single-reference>";
```

```
<single-reference> ::= [<SDO>] <doc#> [( <version> )] [ <R/S/C/T/F> ] [ - <descr> ] | <URL> [ - <descr> ]
```

where

- <SDO>: Standards Defining Organization
 - Examples: IEEE, ITU-T
 - Optional: For well known document types, e.g., RFC implies IETF and TR implies BBF, the SDO is not required.
- <doc#>: Document number
 - Examples: TR-101i2, RFC 7950
 - Mandatory
 - For BBF documents, the Issue and/or Amendment and/or Corrigendum are included in the document number in the abbreviated format as described in IETF-7, e.g., TR-101i2, TR-385i2a1.
- <version>: Version or revision number and/or date
 - Examples: (Revision 12.3), (2008/07)
 - Optional: For documents whose name implies or specifies the version, e.g., RFC 8348, TR-383a6, additional version information is not required.
- <R/S/C/T/F>: Requirement/Section/Clause/Table/Figure
 - Examples: R-13, Section 3.6, Clause 7.2.1, Table 2, Figure 5-4
 - Optional: Only if applicable.
 - Terminology MUST align with the terms used by the reference, e.g., Sections for RFCs, Clauses for ITU-T specifications.
 - First letter is capitalized, e.g., "Section" instead of "section".
 - If multiple are applicable in a given document, treat each as a separate reference.
- <descr>: Description of the reference
 - Examples: variable name or specific identifier within the Section or Clause
 - Optional: Should be used when it is possible to uniquely identify the parameter being modeled within the given reference.
 - Examples:
 - An ME defined in an ITU-T G.988 Clause contains one or more parameters. If the reference is for one specific parameter, it must be identified in the reference, e.g., "ITU-T G.988 Clause 9.9.6 - PSTN protocol variant".
 - A parameter defined in an ITU-T G.997.x Clause is typically fully contained within the clause, i.e., no ambiguity. However, for implementers, it is useful to be able to quickly search for a desired parameter when the Clause is not known. For this use case, both the descriptive and shorthand (if applicable) representation should be identified, e.g., "ITU-T G.997.2 Clause 7.2.1.1 - Maximum net data rate (MAXNDRds/us)".
 - However, if the G.997.x parameter represents more than one distinct value, e.g., upstream and downstream, and the data node only applies to one, the reference should only contain the value for which it is applicable, e.g., "ITU-T G.997.2 Clause 7.2.1.1 - Maximum net data rate downstream (MAXNDRds)".
 - A parameter defined in a Table in SFF-8472 is specified by its memory location (address and byte numbers). References for these types of parameters should include the address and bytes, e.g., "SFF-8472 (Revision 12.4) Table 4-1 - Address A0h, Bytes 96-127".

- In general, the title associated with the reference should not be used unless it also represents one of the above.
- <URL>: The URL of the associated reference used when there is no formal technical specification
 - Example: `github.com/grpc/grpc/blob/master/doc/connection-backoff.md`
 - Do not include the URL scheme, e.g., `http://` or `https://`, as it may change. For example, a site may move from an unsecure (`http`) scheme to a secure (`https`) scheme.
- `;`: Requirement separator
 - Mandatory: A semicolon **MUST** be used to terminate all but the last reference when there are multiple references in a reference statement.
 - Each reference **MUST** start on a new line.
 - This is not applicable when there is only one reference in the reference statement.
- If a single reference spans multiple lines, the first character on the second and subsequent lines of the reference **SHALL** align under the first character on the first line of the reference.

Example:

reference

```
"RFC 5519 Section 5 - mgmdRouterInterfaceQueryInterval;
TR-101i2 Table 2;
SFF-8472 (Revision 12.4) Table 9-5 - Address A2h,
Bytes 40-41;
ITU-T G.997.2 Clause 7.11.1.1 - Net data rate (NDRds/us);
ITU-T G.997.2 Clause 7.2.1.1 - Maximum net data rate downstream (MAXNDRds);
IEEE 802.1Q (2018) Clause 20.23.3;
IEEE 802.1Q (2018) Table 20-1 - xconCCMdefect;
github.com/grpc/grpc/blob/master/doc/connection-backoff.md - INITIAL_BACKOFF";
```

Additional Notes:

- While external references must be documented in the 'reference' statement, it is also allowed to refer to them in the description itself.
- This guideline does not apply to a reference statement within a revision statement. Those reference statements have a unique format used for drafts and publication.
- URL references should be periodically reviewed to ensure they still exist or have not been moved.

BBF-26: Body Statements

The order of body statements in BBF YANG modules **SHOULD** follow the order as stated in the body-stmt ABNF grammar defined in RFC 7950 [4] section 14 and as shown in the YANG Module Template defined in RFC 8407 [6] Appendix B. Note that while the ABNF grammar specifies deviation statements follow notifications, the YANG Module Template states "DO NOT put deviation statements in a published module". BBF YANG modules **MUST** follow this stated guideline on deviation statements.

Body Statement Order:

- extension statements
- feature statements
- identity statements
- typedef statements
- grouping statements
- data definition statements
- augment statements
- rpc statements
- notification statements

BBF-27: 'list' and 'leaf-list' Naming and Descriptions

The name of a 'list' or 'leaf-list' data node SHALL be in the singular form. The motivation is that the XML representation repeats the name for each entry.

Example YANG:

```
list traffic-management-profile {
  key name;
  description
    "A traffic management profile.";

  leaf name {
    type string;
    description
      "The name of the profile.";
  }
}
```

Example XML:

```
<traffic-management-profile>
  <name>multicast</name>
</traffic-management-profile>
<traffic-management-profile>
  <name>hsi</name>
</traffic-management-profile>
```

Note in the example YANG, the description for the 'list' node describes a single entry in the list, "A traffic management profile".

The description statement for a 'list' or 'leaf-list' data node SHALL describe what an entry represents rather than describing the entire list. In other words, the description should follow the example above rather than:

```
list traffic-management-profile {
  key name;
  description
    "A list of traffic management profiles.";

  leaf name {
    type string;
    description
      "The name of the profile.";
  }
}
```

BBF-28: Description on 'when' statements

Not everyone reading a YANG model is an expert in YANG and/or XPath. In order to provide assistance in understanding the intent, all 'when' statements SHALL contain a description statement. This description should convey the intent of the 'when' condition in easy to understand language.

Examples:

```
when "derived-from-or-self(hw:class,'bbf-hwt:transceiver-link') {
  description
    "Applicable when the class of hardware component is either a
    transceiver link or another class derived from a transceiver
    link.";
}
```

```

when "not(boolean(..enable))"
  + " or "
  + "(../enable='true')" {
  description
    "If the administrative state of a session is
    supported, the session must be currently
    administratively enabled to reset the session.";
}

```

BBF-29: Use of English Contractions

Descriptions in YANG modules are considered to be formal documentation. Therefore, English contractions, e.g., can't, isn't, etc., SHALL NOT be used in description statements.

BBF-30: Use of Presence statements

A presence statement is a substatement of a container statement. A container defined in YANG without a presence statement exists in the data tree of a datastore only for organizing the hierarchy of data nodes. A container defined with a presence statement assigns its meaning to the existence of the container in the data tree of a datastore.

The use of a presence statement has the following effect: if mandatory data nodes and/or data nodes with a default value are defined within a presence container, then these data nodes will only exist if the container is present in the data store; if the container is not present, then these data nodes will also not be present in the data store. In contrast, a container without a presence statement that has one or more mandatory data nodes or data nodes with a default value will always exist with those data nodes.

Note that for backward compatibility reasons, mandatory data nodes cannot be added to a newer revision of a YANG module. However, mandatory data nodes can be added to a newer revision conditionally, such as by embedding them in a container with a presence statement.

In BBF modules containers shall be defined with a presence statement when:

- the presence of the container itself in the data tree has a meaning; for example, the presence of the container 'l2-dhcpv4-relay' in module bbf-l2-dhcpv4-relay in a data tree enables the DHCP relay function.

```

container l2-dhcpv4-relay {
  presence
    "Enables the management of DHCPv4 message processing at this
    VLAN sub-interface."; << because of this meaning being assigned
  description

```

- the configuration of data nodes is to be conditionally enforced; for example, the reference to a profile is mandatory in the same 'l2-dhcpv4-relay' container referred to above, because from the moment the DHCP relay function is enabled, the means as to how DHCP messages are to be processed must be configured.

```

container l2-dhcpv4-relay {
  presence
    "Enables the management of DHCPv4 message processing at this
    VLAN sub-interface.";
  leaf enable
  leaf profile-ref {
    when "../enable = 'true'" {
      description
        "Applicable only when L2 DHCPv4 Relay Agent functionality

```

```

        is enabled.";
    }
    type leafref {
        path "/bbf-l2-d4r:l2-dhcpv4-relay-profiles/bbf-l2-d4r:l2-"
            + "dhcpv4-relay-profile/bbf-l2-d4r:name";
    }
    mandatory true; << to make this leaf conditionally mandatory

```

BBF-31: Formatting XPath expression arguments of YANG statements

XPath expression arguments of 'must', 'when' and 'path' statements shall follow the following guidelines:

- the first line of the argument **MUST** be on the next line after the keyword with the opening quote using an additional two space indentation compared with that of the keyword.
- if the argument spans multiple lines, then the argument **MUST** be defined by concatenating individual quoted-string statements defined per line.
- these additional lines shall follow the indentation of the first line of the argument where the characters "+" precede a quoted string for which the opening quote of all lines aligns.

Examples:

```

leaf profile-ref {
    when
        "../enabled = 'true'" {
        description
            "...";
        }
    type leafref {
        path
            "/bbf-icmpv6:icmpv6-profiles/bbf-icmpv6:"
            + "icmpv6-profile/bbf-icmpv6:name";
    }
    description
        "...";
}

leaf shaper-name {
    type leafref {
        path
            "/bbf-qos-tm:tm-profiles/bbf-qos-shap:shaper-profile/"
            + "bbf-qos-shap:name";
    }
    must
        'boolean(/bbf-qos-tm:tm-profiles/bbf-qos-shap:shaper-profile'
        + '[bbf-qos-shap:name=current()]/bbf-qos-shap:'
        + 'single-token-bucket)' {
        error-message
            "...";
        description
            "...";
    }
}

```

BBF-32: Naming choices

Choice statements can be augmented with additional case statements and in some cases choice statements may be defined without case statements with the expectation that case statements are to be

added through augmentations from other BBF modules, from modules from other SDOs or from vendor's proprietary modules.

Therefore, the name of choices should not include the names of specific cases and should always be generic in nature, i.e., describing the purpose of the choice and not any specific implementation.

A bad example would be as follows:

```
choice red-or-green {
  mandatory true;
  case red {
    leaf red-reference {
      type red-ref;
      description
        "A reference to red.";
    }
  }
  case green {
    container green-definition {
      ...
    }
  }
}
```

If the choice above would be augmented with a case 'blue', then the name of the choice would become inconsistent and misleading.

A more generic and preferred naming would be as follows:

```
choice color {
  mandatory true;
  case red {
    leaf red-reference {
      type red-ref;
      description
        "A reference to red.";
    }
  }
  case green {
    container green-definition {
      ...
    }
  }
}
```

BBF-33: Mapping of alarm type identifiers to ITU-T X.733 and ITU-T X.736 event types

ITU-T X.733 Clause 8.1.1 [8] and ITU-T X.736 Clause 8.1.1 [9] define a series of 'event types', such as 'communications alarm type', 'quality of service alarm type' or 'integrity violation' to categorize alarms. Event types are refined through 'event information', which includes a 'probable cause', a 'specific problem' and a 'perceived severity'.

BBF Alarm Management for Access Nodes [1] is based on the YANG module ietf-alarms as defined by RFC 8632 "YANG Data Model for Alarm Management" [Z]. In RFC 8632 an alarm is uniquely identified through an 'alarm type', which consists of an 'alarm type identifier' and an 'alarm type qualifier', and the resource

which has raised the alarm. To be able to be integrated into the BBF Alarm Management, BBF YANG data models must define 'alarm type identifiers'.

To retain continuity for operators between alarm management according to RFC 8632 and the ITU-T X.733 and ITU-T X.736 alarm standards, RFC 8632 supports the additional YANG module ietf-alarms-x733, which augments the alarm inventory, the alarm lists, and the alarm notification with X.733 and X.736 parameters as well as the mapping of event types and probable cause to individual alarm types as summarized in the YANG tree below (alarm notification not shown for clarity as this is basically the same as the entry in the alarm list).

```

+--rw alarms
  +--rw control
  | ...
  | +--rw alarm-shelving {alarm-shelving}?
  | | ...
  | +--rw x733:x733-mapping* [alarm-type-id alarm-type-qualifier-match] {configure-x733-
mapping}?
  |   +--rw x733:alarm-type-id                al:alarm-type-id
  |   +--rw x733:alarm-type-qualifier-match    string
  |   +--rw x733:event-type?                   event-type
  |   +--rw x733:probable-cause?               uint32
  |   +--rw x733:probable-cause-string?       string
  +--ro alarm-inventory
  | +--ro alarm-type* [alarm-type-id alarm-type-qualifier]
  |   +--ro alarm-type-id                alarm-type-id
  |   +--ro alarm-type-qualifier          alarm-type-qualifier
  |   +--ro resource*                    resource-match
  |   ...
  |   +--ro x733:event-type?              event-type
  |   +--ro x733:probable-cause?          uint32
  |   +--ro x733:probable-cause-string?   string
  +--ro summary {alarm-summary}?
  | ...
  +--ro alarm-list
  | ...
  | +--ro alarm* [resource alarm-type-id alarm-type-qualifier]
  | | +--ro resource                resource
  | | +--ro alarm-type-id            alarm-type-id
  | | +--ro alarm-type-qualifier      alarm-type-qualifier
  | | ...
  | | +--ro x733:event-type?          event-type
  | | +--ro x733:probable-cause?      uint32
  | | +--ro x733:probable-cause-string? string
  | | +--ro x733:threshold-information
  | | | +--ro x733:triggered-threshold? string
  | | | +--ro x733:observed-value?      value-type
  | | | +--ro (threshold-level)?
  | | | | +--:(up)
  | | | | | +--ro x733:up-high?          value-type
  | | | | | +--ro x733:up-low?           value-type
  | | | | +--:(down)
  | | | | | +--ro x733:down-low?          value-type
  | | | | | +--ro x733:down-high?         value-type
  | | | +--ro x733:arm-time?             yang:date-and-time
  | | +--ro x733:monitored-attributes* [id]
  | | | +--ro x733:id                   al:resource
  | | | +--ro x733:value?               string

```

```

| | +--ro x733:proposed-repair-actions* string
| | +--ro x733:trend-indication?          trend
| | +--ro x733:backedup-status?           boolean
| | +--ro x733:backup-object?             al:resource
| | +--ro x733:additional-information* [identifier]
| | | +--ro x733:identifier string
| | | +--ro x733:significant? boolean
| | | +--ro x733:information? string
| | +--ro x733:security-alarm-detector? al:resource
| | +--ro x733:service-user?             al:resource
| | +--ro x733:service-provider?         al:resource
| | ...
+--ro shelved-alarms {alarm-shelving}?
| ...
+--rw alarm-profile* [alarm-type-id alarm-type-qualifier-match resource] {alarm-profile}?
...

```

Although it would be possible to define a set of abstract alarm type identities according to X.733 and X.736 event types, this would be problematic, if the hard-coded categorization does not match the operator's own requirements and would not subsequently be able to be modified by operators. The example below shows a possible hierarchy based on X.733 event types for the ANCP YANG data model:

```

import bbf-alarm-types {
  prefix bbf-alt;
}
identity ancp-alarm-type-id {
  base bbf-alt:bbf-alarm-type-id;
}
identity communications-alarm {
  base bbf-alt:ancp-alarm-type-id;
}
identity adjacency-failure {
  base bbf-alt:communications-alarm;
}

```

Therefore, when defining new alarm type identifiers, BBF YANG data models SHOULD NOT define these alarm type identifiers within a fixed hierarchy of alarm types as shown above, but implement the module `ietf-alarms-x733` to support the explicit mapping of alarm types to event types and probable cause. The example below shows the preferred hierarchy as implemented for the BBF ANCP YANG data model:

```

import bbf-alarm-types {
  prefix bbf-alt;
}
identity ancp-alarm-type-id {
  base bbf-alt:bbf-alarm-type-id;
}
identity adjacency-failure {
  base bbf-alt:ancp-alarm-type-id;
}

```

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 a *min-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* [3] to explicitly enforce how many interfaces are present.

```
deviation "/if:interfaces" {
  description
    "There are always 8 G.fast interfaces present.";
  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 a *list* 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.

DEV-7: add - Mandatory leaf

The deviate 'add' argument MAY be used to make an optional leaf mandatory.

A leaf being defined without a 'mandatory true' statement is by definition optional. Making the leaf mandatory is accomplished by adding a 'mandatory true' statement where this was previously not defined. For example, take a leafref to an interface:

```
leaf interface {
  type if:interface-ref;
  description
    "References the lower-layer interface.";
}
```

This leaf can be made mandatory by the following deviation

```
deviation "interface" {
  deviate add {
    mandatory true;
  }
}
```

Note that this is similar to the deviation of introducing a min-elements to a list or leaf-list that also makes one or more elements in the list mandatory.

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.