IEEE P802.11
Wireless LANs

|  |
| --- |
| MIB TruthValue usage patterns |
| Date: 2015-07-12 |
| Author(s): |
| Name | Company | Address | Phone | email |
| Mark Hamilton | Ruckus Wireless | 350 W. Java DrSunnyvale, CA | +1 303 818 8472 | mark.hamilton@ruckuswireless.com |
|  |  |  |  |  |

Abstract

This document contains a description of “design patterns” for the more common usage of MIB attributes with Type TruthValue, in Std 802.11 and its amendments.

R0 – Initial discussion document.

R1 – Fixed typos

R2 – Updates based on face-to-face review: Add examples, Merge 3.1 and 3.2, miscellaneous

R3 – Updates based on face-to-face review in Vancouver (May 2015). Merge both Dynamic attribute types into one pattern, with variants to indicate the differences.

# Introduction and Purpose

This document outlines several common usage models for a subset of MIB attributes: those with data type TruthValue (“SYNTAX TruthValue” in the MIB object definition). Typically, such an attribute is used to indicate the status a feature or a set of behaviors, which either is or is not operational within a given implementation at a given time.

As with all MIB attributes, the benefit of these attributes to the Standard is to provide a model of expected behavior and interactions for implementations of the Standard. Since the MIB is rarely used, literally as defined, by implementation, instead it serves to provide a common definition style and a bit of formalism to descriptions of implementation behavior that is necessary for interoperability. In this regard, the MIB is similar to the service definitions in clause 6 (Layer management), and in fact through the mapping described in subclause 6.2 (Generic management primitives) the MIB attributes indirectly define part of the management service interface.

In this document, only MIB attributes defined with type (SYNTAX) of’”TruthValue” are addressed, as these attributes have the most commonality in purpose, while having considerable variation in naming and definition style for the same uses. It is hoped that with a common set of guidelines for naming and definition style, that all such MIB attributes can (probably over a period of time) be described with a small number of recognizable patterns, and result in ease of understanding their intent.

# Elements of attribute definition, and pattern uniqueness

Each usage pattern below is intended to completely cover the scenario for a given feature. That is, a given feature shall use exactly one of these patterns, so it shall never need or use more than one of these patterns. If a feature scenario is found that does not fit any pattern, or needs more than one pattern, then that should be discussed, and a new pattern for the scenario created if that is necessary.

Each usage pattern below includes guidelines for the following aspects of definitions for MIB attributes that fit that pattern:

* Name – using a consistent set of suffixes on attribute names will help the reader intuitively understand the purpose of the attribute, and thereby the behavior(s) to expect from implementations.
* MAX-ACCESS – this aspect should provide clarity about access to the attribute from an external entity (usually a management interface or system, such as SNMP or similar).
* DESCRIPTION – document 11-09/533 provides guidelines for general MIB attribute definition, including a discussion of the information that should be included. This document provides more specific guidelines specifically for TruthValue attribute patterns listed here.

Each usage pattern also includes guidelines for using and referencing the MIB attribute elsewhere in the Standard.

For the purposes of this document, the term “feature” applies to any identifiable unique feature of the Standard that could be independently present or absent in a particular implementation, or a similar set of behaviors which might be operational as a group, or none of them are.

# Patterns

## dot11<XXX>Implemented: Static implementation capability

A static implementation pattern is for a feature that is an inherent capability of a given implementation. As an “inherent” capability, this pattern is for features which are permanently operational in implementations that support it – that is, it is not enabled or disabled dynamically during the lifetime of an instance of the implementation.

There are two forms of this pattern: internal use only, and externally accessible, as described below

### Internal use only

This form of the static implementation pattern is for a feature that is an inherent capability of a given implementation, and which is not expected to be queried by an external entity. The purpose of such an attribute is really only internal to the 802.11 Standard; defining such an attribute makes it clear that the indication of this support is in fact only useful to the internal 802.11 entities, and in effect becomes just a shorthand formalism (and makes for easier searching, etc.) for “devices that implement XXX” for use elsewhere in the Standard.

### External access provided

The intent of this form of the static implementation pattern is for a feature that is an inherent capability of a given implementation, and where it would be useful for this attribute to be queried (for support in the implementation) by an external entity. Such an attribute can be used within the Standard to control protocol or behaviors which are optional dependent on whether the implementation supports the feature, as well as to inform external management systems of support for the feature thus allowing such systems to manage aspects of the feature, or make other dynamic decisions within the management of the overall deployment.

### Form of definition and use

Both forms of this pattern have similar definition, only the setting for MAX-ACCESS differs, and the use in the Standard is also similar.

Name: dot11<XXX>Implemented

MAX-ACCESS: none - access to external entity not allowed

 OR

MAX-ACCESS: read-only - access to external entity allowed

DESCRIPTION: "This is a capability variable. Its value is determined by device capabilities. This attribute, when true, indicates that the XXX feature is implemented and operational. This attribute, when false or not present, indicates that the XXX feature is not implemented or not operational."

The attribute can then be referenced in the body of the Standard as a quick indication of the presence or absence of the feature in an implementation, for example:

- for parameters to service primitives in clause 6, “This parameter is present if dot11<XXX>Implemented is true.”

- for optional fields with frame formats in clause 8, “The <optional field name> is present if dot11<XXX>Implemented is true.”

- for description of behavior in later clauses and Annexes, “If dot11<XXX>Implemented is true, <some behavior happens>.”

### Example

The MIB attribute dot11RSNAOptionImplemented (as used in IEEE Std 802.11-2012) is an example of an attribute that should use this pattern. There is no indication (in IEEE Std 802.11-2012) that this attribute has any purpose for external access (an external entity reading its state). So, it seems it could/should have MAX-ACCESS of “none”. However, it is shown as “read-only” in that version of the Standard. There should either be a description of how or when such access is useful, or the access should be changed to “none”.

The resulting example, applying the conventions above, would be:

dot11RSNImplemented OBJECT-TYPE

SYNTAX TruthValue

MAX-ACCESS none

STATUS current

DESCRIPTION "This is a capability variable. Its value is determined by device capabilities. This attribute, when true, indicates that RSN is implemented and operational. This attribute, when false or not present, indicates that RSN is not implemented or not operational.”

::= { dot11StationConfigEntry 26 }

## dot11<XXX>Activated: Dynamically operational capability

### General

This pattern is for a feature that, when present in an implementation, becomes operational or non-operational dynamically within the lifetime of a particular instance of the implementation. Such dynamic changes occur as a result of behaviors or interactions described within Std 802.11, for example, based on a protocol exchange, or receiving an enablement indication from a peer entity, or as a result of an external entity writing to the MIB attribute. It is critical to unambiguous description of the behavior that only one entity be able to change the attribute, whether that is an internal or external entity.

Such an attribute can be used within the Standard to control protocol or behaviors which are

dependent on whether the feature is currently operational, as well as to both allow an external entity to change the operational state as well as to inform an external entity of the current operational state of the feature thus allowing such systems to manage aspects of the feature, or make other dynamic decisions within the management of the overall deployment.

The 802.11 Standard must describe the change in behavior of a conforming system. If an external entity can modify the state, this adds the complexity of describing the behavior when an external entity changes the attribute state at arbitrary times. This response to an externally written change may include delaying any change in behavior until a later time or trigger event has occurred. If there are constraints on when the attribute can be changed, those must be described as an implementation requirement to enforce such limitations, to prevent unspecified behavior.

### Form of definition and use

The form of definition depends on whether an internal or external entity can write to the attribute, and whether the attribute is made available for query by an external entity.

Name: dot11<XXX>Activated

MAX-ACCESS: none - access to external entity not allowed, and written by internal entity

MAX-ACCESS: read-only - query of state by external entity allowed, but written by internal entity

 OR

MAX-ACCESS: read-write - modification of state by external entity allowed, query of state by external entity is always also allowed

It is written by <some internal entity> when <some defined event happens>.."

 OR

DESCRIPTION: "This is a control variable. It is written by an external management entity. This attribute, when true, indicates that the XXX feature is currently operational. This attribute, when false or not present, indicates that the XXX feature is currently not operational. Changes take effect when <some defined event happens>."

The attribute can then be referenced in the body of the Standard as a quick indication of the current operational state of the feature, for example:

- for parameters to service primitives in clause 6, “This parameter is present if dot11<XXX> Activated is true.”

- for optional fields with frame formats in clause 8, “The <optional field name> is present if dot11<XXX> Activated is true.”

- for description of behavior in later clauses and Annexes, “If dot11<XXX>Activated is true, <some behavior happens>.”

### Examples

dot11ExtendedChannelSwitchActivated OBJECT-TYPE

DESCRIPTION "This is a control variable. It is written by the SME when the device is initialized for operation in a band defined by an Operating Class. This attribute, when true, indicates that the station implementation is capable of supporting Extended Channel Switch Announcement. This attribute, when false or not present, indicates the capability is currently not operational

87 }

dot11RSNAProtectedManagementFramesActivated OBJECT-TYPE

SYNTAX TruthValue

MAX-ACCESS read-write

STATUS current

DESCRIPTION "This is a control variable. It is written by an external management entity.This variable indicates whether this STA enables management frame protection."

DEFVAL { false }

::= { dot11StationConfigEntry 88}

dot11SpectrumManagementRequired is an example of an attribute set both internally as well by an external management entity. The internal use is implied, as a STA must set this to true (if it isn’t already set to true by a management entity) before it can associate to a BSS that is advertising it.

dot11SpectrumManagementActivated OBJECT-TYPE

SYNTAX TruthValue

MAX-ACCESS read-write

STATUS current

DESCRIPTION "This is a control variable. It is written by the SME or external management entity. Changes take effect for the next MLME-START.request primitive. A STA uses the defined TPC and DFS procedures if this attribute is true; otherwise it does not use the defined TPC and DFS procedures."

DEFVAL { false }

::= { dot11StationConfigEntry 25 }