IEEE P802.11
Wireless LANs

|  |
| --- |
| MIB TruthValue usage patterns |
| Date: 2017-07-31 |
| Author(s): |
| Name | Company | Address | Phone | email |
| Mark Hamilton | Ruckus/Brocade | 350 W. Java DrSunnyvale, CA | +1 303 818 8472 | mark.hamilton2152@gmail.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.

R4 – Updates for consideration at face-to-face in San Antonio (Nov 2016).

R5 – Start at adding “Pattern C”, “Pattern D” and “Usage Z”, work in progress, needs discussion.

R6 – Finished proposal for adding “Pattern C”, “Pattern D” and “Usage Z”. Needs discussion.

# 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 (“Pattern A”)

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 that are permanently operational in an instantiation of an implementation that supports 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 (“Pattern B”)

### 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 current state of the feature’s operational state may or may not be made available to query by an external entity.

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

 OR

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

DESCRIPTION: "This is a status variable. It is written by <some internal entity> when <some defined event happens>. 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."

 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

SYNTAX TruthValue

MAX-ACCESS read-only

STATUS current

DESCRIPTION "This is a status 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."

DEFVAL { false }

::= { dot11StationConfigEntry 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. Changes take effect as soon as practical in the implementation. This variable indicates whether this STA enables management frame protection."

DEFVAL { false }

::= { dot11StationConfigEntry 88}

## dot11<XXX>Required: Capability controlled by master/slave relationship (“Pattern C”)

### General

This pattern is for a feature that is required to be operational within a ‘slave’ device, as indicated by a ‘master’ (such as an AP, or external database). The operational requirements for the feature, and the method of communication from master to slave, are described with Std 802.11. The feature will be statically operational for the lifetime of the master instantiation. The feature will be operational within the slave at least for the lifetime of the master/slave relationship.

Such an attribute can be used within the Standard to control protocol or behaviors which are dependent on whether the feature is currently operational on the master and/or slave, ~~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 current state of the feature’s operational state may or may not be made available to query by an external entity.

In addition to describing the behavior of both a master and slave when the feature is operational, the 802.11 Standard must describe the behavior of a conforming slave system when the feature transitions between operational or not operational, and the method of interaction between a master and slave.

Used on slave to control which master it will associate.

### 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>Required

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

 OR

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

DESCRIPTION: "This is a master/slave variable. Its value on <a master device> is determined by <regulatory requirements, local conditions, etc.>. Its value on <a slave device> is determined by the <relationship to> <a master device>. 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."

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> Required is true.”

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

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

### Examples

dot11SpectrumManagementRequired OBJECT-TYPE

SYNTAX TruthValue

MAX-ACCESS read-write

STATUS current

DESCRIPTION "This is a master/slave 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 }

## dot11<XXX>Policy: Capability controlled by external policy control and not signaled (“Pattern D”)

### General

This pattern is for a feature that becomes operational or non-operational dynamically within the lifetime of a particular instance of the implementation, but is only enabled by external policy, and is not signaled over the air to peers.

Such an attribute can be used within the Standard to control protocol or behaviors which are dependent on whether the feature is currently operational, under the control of an external entity.

### Form of definition and use

The form of definition is as shown below.

Name: dot11<XXX>Policy

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

DESCRIPTION: "This is a policy variable. 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."

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> Policy is true.”

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

### Examples

dot11OperatingClassesPolicy OBJECT-TYPE

SYNTAX TruthValue

MAX-ACCESS read-write

STATUS current

DESCRIPTION "This is a control variable. It is written by an external management entity. Changes take effect for the next MLME-START.request primitive. A STA uses the defined operating classes procedures if this attribute is true."

DEFVAL { false }

::= { dot11StationConfigEntry 29}

dot11RSNAPBACPolicy OBJECT-TYPE

SYNTAX TruthValue

MAX-ACCESS read-write

STATUS current

DESCRIPTION "This is a control variable. It is written by an external management entity. Changes take effect as soon as practical in the implementation. This variable indicates whether this STA requires the Protection of block ack agreements."

DEFVAL { false }

::= { dot11StationConfigEntry 93}

## No MIB entry, use words (“Usage Z”)

### General

This is not a MIB pattern, but is a categorization a feature that does not need a MIB entry. Such a feature is generally referenced in a very small number of places, and can therefore be referenced with simple wording within the body of the Standard, without undue complexity or any ambiguity.

Such a feature is not controllable by an external entity, and is static for the lifetime of an instantiation of the entity. <Example, dot11ImmediateBlockAckOptionImplemented>

OR

Such a feature is either not controllable by an external entity, or any such control is not standardized and is implementation dependent. <Example, dot11MSGCFActivated>

### Form of definition and use

There is no MIB definition for these features.

In the body of the Standard, it’s (rare) references will appear with descriptive text.

### Examples

“A STA sets the Immediate Block Ack subfield to 1 within the Capability Information field when the station implementation is capable of supporting immediate block ack and sets it to 0 otherwise.”

“The MSGCF Capability field is set to 1 to indicate the non-AP STA has been set by an external management entity or the SME as capable of providing the MSGCF in 6.4.”

**Open Items for consideration:**

**Item 1:**

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 }

Per Pattern C, above, this is because this is a “mater/slave” attribute. Needs discussion whether this should be one attribute (with different behavior on the master versus the slave, but otherwise a lot of overlap, too), or two separate attributes.

**Item 2:**

* dot11MultiDomainCapabilityImplemented
* dot11MultiDomainCapabilityActivated

 – Is there value in having both of these? This is only one of many examples where the \*Implemented attribute does not seem to have any added value, and it is not clear why the \*Activated attribute would ever change during operation.

Tentative agreement: No pattern for these. The feature can be classified as Pattern A or Pattern B, and never needs both.

**Item 3:**

* dot11SpectrumManagementImplemented
* dot11SpectrumManagementRequired

- Definitely an “Implemented/Activated” type of thing, but not spelled like one

- Again, is there value in both of these?

Tentative agreement: No pattern for these. The feature can be classified as Pattern A or Pattern B (with that Pattern’s spelling), and never needs both.

**Item 4:**

* dot11AssociateInNQBSS

- Definitely an “Implemented/Activated” type of thing, but not spelled like one.

- Never used outside the MIB. Is that useful? Should it be handled some other way?

Tentative agreement: Obsolete or deprecate. Not useful.

**Item 5:**

* dot11DLSAllowedInQBSS

 - Definitely an “Implemented/Activated” type of thing, but not spelled like one. Usage (and relation to QoS) is pretty confusing. Is it ever signalled? How does “If … direct streams are allowed in the policy of the BSS (as determined by dot11DLSAllowedInQBSS),” become known to STAs?

Tentative agreement: In this case, at least, it can be Usage Z: no need for a MIB attribute at all, just describe in text using words.

**Item 6:**

* dot11TxAntennaImplemented

 – Never used except in DSSS and ERP PHY characteristics.

**Item 7:**

MIB attributes of the form “\*Implemented” that are not of type TruthValue and some notes on them. They may reflect more information, such as “how much/many of X is implemented?”

* dot11RSNAConfigPairwiseCipherImplemented OCTET STRING
* dot11RSNAConfigPairwiseCipherSizeImplemented Unsigned32
	+ Above used in dot11RSNAConfigPairwiseCiphersTable, which is never referenced. The MIB claims this is used by an external management entity: “The pairwise cipher suite list in the RSNE is formed using the information in this table.” How this is accomplished appears to be missing.
* dot11TVHTMUMaxUsersImplemented – Never used except in the MIB
* dot11WEPKeyMappingLengthImplemented – OK, this is an implementation imposed limit and discussed in text, but doesn’t affect over-the-air signaling.
* dot11NumberSupportedPowerLevelsImplemented – Never used except in the DSSS, HR and HT PHY characteristics

**Item 8:**

Are there three concepts - are all three necessary/useful/relevant to the scope of the Standard: “hard-wired/manufactured ‘capable’”, “’enabled’, by something/someone, at say, power on”, and “’activated’ dynamically”?

Tentative agreement: Only two useful concepts: static during an instantiation (Pattern A), and may change during instantiation (Pattern B). Agreed that the difference between “hard-wired/manufactured” that way, and “instantiated/powered-on, and can’t change” that way, is not useful.

To be discussed: Is an “instantiation” from MLME-START.request to MLME-START.request? Or, is it longer than that? Clearly, it is longer than each MLME-JOIN.request.

**Item 9:**

There are examples of \*Activated where the change takes effect “as soon as practical”, and examples where the change takes effect at the next MLME-Start or MLME-JOIN. (And, examples where it is not specified at all… ☹ ) Are both the first two types actually meaningful/useful in the Standard?

**Item 10:**

Is there a difference between an externally set (for example, at initialization) control over the activation/enablement of a feature, and an externally reported but not necessary for interoperability piece of information? Example, supported rates versus 11k counters.

Tentative agreement: Yes, these are different. The important distinction is when something can change. A status reported and changing attribute (like 11k counters) are different from a static feature.

To be discussed: Is there an important difference between a reported status (like 11k counters) and a slow-changing, but dynamically changeable control, like dot11RSNAActivated? Are these both the same pattern (Pattern B), or do we want separate patterns?

**Background/Historical discussion:**

*If there is only dot11XxxActivated (not dot11XxxImplemented), can the external management entity try to read it, and determine if it is implemented by whether that read returns with an error or not? How about if it tries to write to it – same thing?*

From RFC 1157 (SNMP):

 (1) if said variable is defined in the MIB with "Access:" of

 "none," it is unavailable as an operand for any operator;

 (2) if said variable is defined in the MIB with "Access:" of

 "read-write" or "write-only" and the access mode of the

 given profile is READ-WRITE, that variable is available

 as an operand for the get, set, and trap operations;

 (3) otherwise, the variable is available as an operand for

 the get and trap operations.

Upon receipt of the GetRequest-PDU, the receiving protocol entity

 responds according to any applicable rule in the list below:

 (1) If, for any object named in the variable-bindings field,

 the object's name does not exactly match the name of some

 object available for get operations in the relevant MIB

 view, then the receiving entity sends to the originator

 of the received message the GetResponse-PDU of identical

 form, except that the value of the error-status field is

 noSuchName, and the value of the error-index field is the

 index of said object name component in the received

 message.

The SetRequest-PDU has a very similar rule, also showing the return of a noSuchName error.

It seems that, yes, we can assume a dot11XxxActivated attribute will return an error upon either read (get) or write (set) operations, if the device does not implement the Xxx feature.

*“MAX\_ACCESS” in MIB: what does this mean? Access given to which entity (any entity other than the “owner”)? Do the SNMP RFCs give any guidance?*

From RFC 2578 (one of the SMIv2 RFCs):

Mapping of the MAX-ACCESS clause:

The MAX-ACCESS clause, which must be present, defines whether it

 makes "protocol sense" to read, write and/or create an instance of

 the object, or to include its value in a notification.

The value of the MAX-ACCESS clause for objects with a SYNTAX clause

 value of Counter32 is either "read-only" or "accessible-for-notify".

In general, the SMI and SNMP RFCs seem to use the “ACCESS” (and “MAX-ACCESS” and “MIN-ACCESS”) clause in reference to the type of access provided by the “agent”, and the “agent” is generally the device (or that portion of the device) that provides SNMP access to the MIB.