IEEE P802.11  
Wireless LANs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Issues with MIB TruthValue usage patterns | | | | |
| Date: 2015-May-11 | | | | |
| Author(s): | | | | |
| Name | Company | Address | Phone | email |
| Brian Hart | Cisco Systems |  |  | brianh@cisco.com |
|  |  |  |  |  |

Abstract

This document describes issues observed with 15/355 which contains a description of “design patterns” for the more common usage of MIB attributes with Type TruthValue, in Std 802.11 and its amendments.

Rev 0: Initial version

Rev 1: Updates based on offline discussion

# Discussion

**Major Points**

1. Seeking agreement that “implementation” and “instance of implementation” are not quite the same, including for APs. As a corollary
   1. Seeking agreement for more precision on use of “implementation” and “instance of implementation”
2. Seeking agreement that “implemented”, “capability” and “capability of support” are synonyms
3. Seeking agreement that “implemented” is not a synonym of “support”. As a corollary
   1. Seeking agreement for more precision on use of “support”
4. Seeking agreement that the 802.11 standard not only defines the over the air protocol, but it also addresses aspects of how a system vendor[[1]](#footnote-2) interacts with their component vendor[[2]](#footnote-3), and these interaction paths should be maintained
5. Seeking agreement that a system vendor has the right to disable any non-fundamental feature (think mandatory features in 802.11-2003) even if their component vendor implements the feature
6. Seeking agreement that the most natural way for a system vendor to disable an implemented feature is via an Activated MIB variable. As corollaries, seeking agreement that:
   1. The system vendor should be able to programmatically determine which features are implemented and thereby available for activation / deactivation
   2. Activated MIB variables need a (shadow?) Implemented MIB variable
7. Seeking agreement that “implemented” is (should be) related to what the component vendor provides, and is expressed by Implemented MIB variables
8. Seeking agreement that “support” is (should be) related to what the system vendor enables, and is expressed by Activated MIB variables

***Item 1: Seeking agreement that “implementation” and “instance of implementation” are not quite the same, including for APs.***

Implementation represents the underlying HW and SW version(s). Instance of an implementation, as reported in 15/355, refers to the implementation beginning after a MLME-RESET (and presumably ending at the next MLME-RESET).

The difference matters since the MLME-RESET primitive has the parameter SetDefaultMIB. For MLME-RESET(SetDefaultMIB= true), this is a “full” reset since “all MIB attributes are set to their default values. The default values are implementation dependent.” That is, the properties of the instance of an implementation come from the underlying properties of the implementation (i.e., two different concepts).

Furthermore, there is also a “partial reset”, from when MLME-RESET(SetDefaultMIB= false) is used instead. Here the MIB variables are not changed. In this case, the new instance retains all *modified* read-write MIB variables, and is thereby this instance of the implementation is notably divorced from the underlying implementation.

SetDefaultMIB cannot be avoided / deprecated by requiring it to always be true since dot11OCBActivated has use cases when SetDefaultMIB is false.

BTW, for an AP, we can create the following representation of the various lifetimes:

A diagram of a computer

AI-generated content may be incorrect.

For a non-AP STA, the following representation applies

A diagram of a diagram

AI-generated content may be incorrect.

Features can survive across JOINs, etc.

***Item 1a: Seeking agreement for more precision on use of “implementation” and “instance of implementation”***

Follows directly from item 1.

For instance, for Implemented MIB variables, the MIB language template should not be (for instance):

* “The MIB variable, if true, indicates that the implementation is capable of supporting …”

… and should rather be …

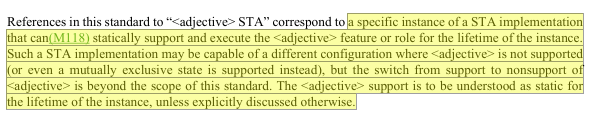
* “The MIB variable if true indicates that the implementation, from the most recent MLME-RESET(SetDefaultMIB = true), is capable of supporting …”

***Item 2: Seeking agreement that “implemented”, “capability” and “capability of support” are synonyms***

As it says (see also item 3) … (and, if so, let’s document this)

***Item 3: Seeking agreement that “implemented” is not a synonym of “support”.***

As useful background, let’s review what we mean by “<adjective> STA”



That is:

* Implementations have capability
* An implementation may support or not support a feature or role, according to configuration
* Support / nonsupport is static for the lifetime of the instance, unless there is explicit language otherwise
* A switch from support to nonsupport is possible, so then “support” has a shorter lifetime than “implementation”.

Accordingly, the definition of “<adjective> STA” expresses notions of:

* Not support: Implementation (aka capability; also 802.11 uses “capability of support” which is consistent with this terminology)
* Static support for the lifetime of the instance
* Nonstatic support during the lifetime of the instance (given explicit discussion). From experience we know there are practically important sub-cases such as:
  + For the lifetime of the BSS (START till STOP) or “association” (but really from JOIN until … actually I don’t know but presumably something after deauth)
  + A portion of the lifetime of a BSS/association, such as “as soon as practically possible”

***Item 3a: Seeking agreement for more precision on use of “support”***

Follows directly from item 3.

Specifically, implementations have “capability” of a feature or “capability of support” of a feature, not “support” for a feature

***Item 4: Seeking agreement that the 802.11 standard not only defines the over the air protocol, but it also addresses aspects of how a system vendor interacts with their component vendor, and these interaction paths should be maintained***

For instance we have the PICS. This is how a component vendor describes the capabilities of their product to a) interested system vendors and/or b) interested end customers[[3]](#footnote-4) of system products.

For instance, consider the MLME/PLME interface provided to SME/EMS.

* The SME to MLME interface is useful for expressing how upper layers can/should use the 802.11 subsystem
* The SME/EMS to MLME interface is useful for describing how a feature be changed from implemented to supported (and especially the Implemented and Activated MIB variables; see below).

***Item 5: Seeking agreement that a system vendor has the right to disable any non-fundamental feature (think mandatory features in 802.11-2003) even if their component vendor implements the feature***

This should be non-controversial.

For instance, if the implementation is poor or an interoperability issue is discovered with the feature, then the system vendor might choose to disable the affected feature.

***Item 6: Seeking agreement that the most natural way for a system vendor to disable an implemented feature is via an Activated MIB variable.***

An Implemented MIB variable (read-only) with an underlying default can be implemented as an Activated MIB variable (read-write) that is written by an SME/EMS to the same value as that underlying default. Therefore, the distinctive value of an Implemented MIB variable is that it is known a priori by the implementation itself; and so does not need knowledge / storage at an SME/EMS. Conversely, if Implemented MIB variables did not have a meaningful underlying defaultthen Implemented variables would have no distinctive value and should not exist.

Accordingly, the default for an Implemented MIB variable is naturally owned by the component vendor. Then, if the system vendor or their customer (netadmin) wants to influence their product’s behavior, the most natural approach is for system vendor/netadmin to use *other* MIB variables where here the poster child is the corresponding Activated MIB variable. This is then used to modify which features are enable/disabled.

Sidebar: that is, we are saying that the value of an Activated variable extends beyond whether dynamic support is required or not. Activated variables help clarify the relationship between, and roles of, system and component vendors.

Mapping these ideas to MIB variables, we see three important cases:

A screenshot of a computer screen

AI-generated content may be incorrect.

Caveat: this figure does not talk about in-service patched SW version(s). These don’t seem to be consistent with Implemented variables, or at least any new capability introduced by in-service patching should be deferred until the next MLME-RESET(SetDefaultMIB = true).

***Item 6a: Seeking agreement that the system vendor should be able to programmatically determine which features are implemented and thereby available for activation / deactivation***

A management system cannot manage what it cannot measure. Before trying to activate a feature, it is first necessary to determine if the feature actually exists.

***Item 6b: Seeking agreement that Activated MIB variables need a (shadow?) Implemented MIB variable***

Given 6a, an Activated variable without an Implemented variable is incomplete.

We have two reasonable paths forward:

* Option 1: Define two MIB variables per activatable feature (Implemented and Activated).
  + Next step
    - Change the second pattern to define two MIB variables.
    - Add Implemented MIB variables
  + Pros
    - Follows pre-15/355 consensus
    - This is logically correct and useful for actual implementations that follow the Annex C MIB definition.
  + Cons
    - A lot of work (and text)
    - Not aligned with 15/355
    - From about 11k onwards, there are no known implementations that follow the Annex C MIB definition.
    - .. so a perfect ASN.1 definition for the MIB has little value
* Option 2: Define that each Activated-only MIB variable always has a corresponding Implemented MIB variable, but the Implemented MIB variable might be implied (shadow)
  + Next step
    - Add, say in Annex C.1 (General), “,Each <XXXX>Activated MIB variable that lacks a corresponding <XXXX>Implemented MIB variable is understood to have an implied <XXXX>Implemented MIB variable. This ellsion is possible since the current version of the MIB is not used, literally as defined, by implementations.” Or similar.
  + Pros
    - Roughly aligned with 15/355
    - Although the MIB is left incomplete for (MIB-based) external management systems, it does compress the MIB description which is a tolerable tradeoff given the lack of such (MIB-based) external management systems.
  + Cons
    - An (MIB-based) external management system would need to define additional variables to build a MIB-based workable system, and so this will be a proprietary approach. Given that constraint, it is more likely that the EMS will instead use a more-modern data model and protocol such as Yang + NETCONF or similar. This outcome is consistent with 802.11 not overly investing in the MIB

***Item 7: Seeking agreement that “implemented” is (should be) related to what the component vendor provides, and is expressed by Implemented MIB variables***

See arguments under item 6.

***Item 8: Seeking agreement that “support” is (should be) related to what the system vendor enables, and is expressed by Activated MIB variables***

See arguments under item 6.

To make this items concrete: consider an 11be AP that the netadmin wishes to operate in 11ax mode (e.g., due to security capabilities of client mix). Here the netadmin (via a UI to the AP’s SME/their EMS) first GETs dot11HEOptionImplemented and dot11EHTOptionImplemented to determine that the AP indeed supports HE and EHT. Here both return true. Then the netadmin (via a UI to the AP’s SME/their EMS) SETs dot11HEOptionActivated to true and dot11EHTOptionActivated to false; then MLME-START is performed. The AP starts in HE mode.

However, neither dot11HEOptionActivated nor dot11EHTOptionActivated actually exist because this use case was not really thought through ☹.

This underlines that Implemented and Activated MIB variables bring distinct value, not just because of the lifetime of the parameter that they signal but which entity establishes their values.

**Proposed Changes and Next Steps**

The following is a marked-up version of 15/355r13, raising issues that have been observed. After discussion and consensus, the suggestion is to prepare an r14 of 15/355r13.

# 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, it instead 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 (and takes precedent if/where there is a conflict).

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.

For the purposes of this document, a new “instantiation” begins with each MLME-RESET.request[[4]](#footnote-5). Note, neither MLME-START.request nor MLME-JOIN.request start a new instantiation.

# Patterns

## dot11<XXX>Implemented: Static implementation capability

This section pertains to features that have a dot11<XXX>Implemented MIB variable but not a corresponding dot11<XXX>Activated MIB variable.

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, during an instantiation of an implementation and also across instantiations of the implementation. As a corollary, the static feature 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 and managed by an external entity. The purpose of such an attribute is only internal to the 802.11 Standard; defining such an attribute makes it clear that the indication of this capability of 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 capability of support in the implementation) by an external entity. Such an attribute can be used within the Standard to control protocol or behaviors that are dependent on whether the implementation is capable of support of the feature, as well as to inform external management systems of the implementation’s capability 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 and (actual/implied) dot11<XXX>Implemented: Dynamically operational capability with capability indication

### General

This pattern is for a feature that, when present in an implementation, can become operational or non-operational dynamically within the lifetime of an 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 provide an 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 and 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}

dot11RadioMeasurementImplemented OBJECT-TYPE

SYNTAX TruthValue

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"This is a capability variable.

Its value is determined by (#3375)STA capabilities.

This attribute, when true, indicates that the station implementation is capable of supporting Radio Measurement. Otherwise it is not capable of performing Radio Measurement."

::= { dot11StationConfigEntry 51 }

dot11RadioMeasurementActivated OBJECT-TYPE

SYNTAX TruthValue

MAX-ACCESS read-write

STATUS current

DESCRIPTION

"This is a control variable.

It is written by an external management entity when any of the MIB attributes listed in 9.4.2.43 (RM Enabled Capabilities element) is equal to true.

Changes take effect with the next MLME-START.request primitive or MLME-JOIN.request primitive.

This attribute, when true, indicates that one or more of the Radio Measurement Activated Capabilities MIB attributes, listed in 9.4.2.43 (RM Enabled Capabilities element), are equal to true. A STA may use the defined Radio Measurement procedures if this attribute is true."

DEFVAL { false }

::= { dot11StationConfigEntry 52 }

## dot11<XXX>Required: Static capability controlled by primary/secondary relationship

### General

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

In general, the primary will be the transmitter of the current state or available options for the feature. The secondary might adopt this state, or choose from the options, or might use the transmitted information as part of a selection process (choosing an AP with which to associate, etc.).

Note, the relationship of “primary” and “secondary” is limited in scope to this particular MIB attribute. There may be no general relationship, or there may be other relationships between the devices that contain the STAs involved.

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 primary and/or secondary, as well as to inform an external entity of the current operational state of the feature.

In addition to describing the behavior when operational on a primary and when operational on a secondary, the 802.11 Standard must describe the behavior of a conforming secondary system when the feature transitions between operational or not operational, and the method of interaction between the primary and secondary.

Note that it is likely that the primary and secondary have different behavioural roles to play with respect to the feature, and the text describing the attribute needs to be clear about these roles.

### Form of definition and use

The form of definition depends on whether the attribute is made available for query by an external entity.

Name: dot11<XXX>Required

MAX-ACCESS: none - access by 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

DESCRIPTION: "This is a primary/secondary variable. Its value on <a primary device> is determined by <regulatory requirements, local conditions, etc.>. Its value on <a secondary device> is <describe relationship to> <a primary 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>Required is true on <the primary entity>, <some behavior happens to advertise it>.”

“If dot11<XXX>Required is true on <the secondary entity>, <some behavior happens to choose a primary based on it>.”

“If < an indication is received on a secondary, e.g., a Beacon received from the primary indicates the state>, then dot11<XXX>Required shall be set to <the appropriate required state>, and <some behavior happens>.”

### Examples

dot11SpectrumManagementRequired OBJECT-TYPE

SYNTAX TruthValue

MAX-ACCESS read-write

STATUS current

DESCRIPTION "This is a primary/secondary variable. It is written by the SME or external management entity on an AP or DFS owner. This variable is static on the AP for the lifetime of the BSS. If the AP or DFS owner advertises Spectrum Management is required, a non-AP or peer STA must set this variable to true prior to associating/peering with the AP/DFS owner. 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>Directed: Dynamic capability controlled by primary/secondary relationship

### General

This pattern is for a feature that is required to be operational within a ‘secondary’ device, as indicated by a ‘primary’ (such as an AP, peer device), and is potentially determined by either the primary or a (possibly delegated) secondary, and may change during the lifetime of the primary instantiation. The operational requirements for the feature, and the method of communication between primary and secondary, are described within Std 802.11.

This exists for situations where the secondary device(s) can set the feature to be operational, based on locally-detected conditions, and communicate that information to the primary and/or other secondaries. However, this is valid only if there are logical protections for any race condition between the secondary’s local detection methods and the primary’s indications.

This is also used if the state on the primary can be modified during the instantiation, for example by an external management entity, in which case the text needs to describe when the changes take effect and how they are propagated.

Note, the relationship of “primary” and “secondary” is limited in scope to this particular MIB attribute. There may be no such general relationship, or other relationships between the devices that contain the STAs involved.

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 primary and/or secondary.

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 primary and secondary when the feature is operational, the 802.11 Standard must describe the behavior of a conforming secondary system for detecting/causing state change of the feature, for when the feature transitions between operational or not operational, and the method of interaction between a primary and secondary.

Note that it is likely that the primary and secondary have different behavioural roles to play with respect to the feature, and the text describing the attribute needs to be clear about these roles.

### 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>Directed

MAX-ACCESS: none - access by 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 primary/secondary variable. Its value on <a primary device> is determined by <regulatory requirements, local conditions, management setting, indications from one or more secondaries, etc.>. Its value on <a secondary device> is determined by the <relationship to> <a primary device>, or local conditions. 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> Directed is true.”

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

- for description of behavior in later clauses and Annexes:

“If dot11<XXX>Directed is true on <the primary entity>, <some behavior happens to advertise it>.”

“If dot11<XXX>Directed is true on <the secondary entity>, <some behavior happens to do it>.”

“If <an indication is received on a secondary, e.g., a Beacon received from the primary indicates the state>, then dot11<XXX>Directed shall be set to <the appropriate required state>, and <some behavior happens>.”

“When <a condition is locally detected on the primary, or an indication is received from a secondary> then dot11<XXX>Directed is set to <the appropriate required state> and <advertised to all secondaries>.”

“When <a condition is locally detected on a secondary> then dot11<XXX>Directed is set to <the appropriate required state> and <indicated to the primary>.”

### Examples

dot11FortyMHzIntolerantDirected OBJECT-TYPE

SYNTAX TruthValue

MAX-ACCESS read-write

STATUS current

DESCRIPTION "This is a primary/secondary variable. It is written by the SME or external management entity, or in response to locally detected or communicated conditions. Changes take effect as soon as practical in the implementation. This attribute, when true, indicates that the STA requests or requires that 40 MHz mask PPDUs are not transmitted within range of the STA.

DEFVAL { false }

::= { dot11OperationEntry 33}

## dot11<XXX>PolicyActive: Feature(behavior) controlled by external policy control and not signaled

### 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>PolicyActive

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> PolicyActive is true.”

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

### Examples

dot11OperatingClassesPolicyActive OBJECT-TYPE

SYNTAX TruthValue

MAX-ACCESS read-write

STATUS current

DESCRIPTION "This is a policy 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}

dot11RSNAPBACPolicyActive OBJECT-TYPE

SYNTAX TruthValue

MAX-ACCESS read-write

STATUS current

DESCRIPTION "This is a policy 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

### 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. An example (from 802.11-2016) is dot11ImmediateBlockAckOptionImplemented.

OR

Such a feature is controllable by an external entity, but any such control is not standardized and is implementation dependent. An example (from 802.11-2016) is dot11MSGCFActivated.

Note, the examples above may not appear in Std 802.11 after 2016, if they are removed per this recommendation.

### Form of definition and use

There is no MIB definition for these features.

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

### Examples

Change from:

To:

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.

A STA sets the Immediate Block Ack subfield to 1 within the Capability Information field when dot11ImmediateBlockAckOptionImplemented is true and sets it to 0 otherwise.

(with MIB definition:)

dot11ImmediateBlockAckOptionImplemented OBJECT-TYPE

SYNTAX TruthValue

MAX-ACCESS read-only

STATUS current

DESCRIPTION "This is a capability variable. Its value is determined by device capabilities. This attribute, when true, indicates that the station implementation is capable of supporting immediate block ack. The capability is disabled, otherwise."

DEFVAL { false }

::= { dot11StationConfigEntry 31}

Change from:

When dot11MSGCFActivated is true, the MSGCF Capability field is set to 1 to indicate the non-AP STA supports the MSGCF in 6.4. When dot11MSGCFActivated is false, the MSGCF Capability is set to 0 to indicate the non-AP STA does not support this capability. APs set this field to 0.

(with MIB definition:)

dot11MSGCFActivated OBJECT-TYPE

SYNTAX TruthValue

MAX-ACCESS read-write

STATUS current

DESCRIPTION "This is a control variable. It is written by an external management entity or the SME. Changes take effect as soon as practical in the implementation. This attribute, when true, indicates the capability of the non-AP STA to provide the MSGCF is enabled. The capability is disabled, otherwise. The default value of this attribute is false."

DEFVAL {false}

::= { dot11StationConfigEntry 130 }

To:

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.

# Recommendations

* Use the patterns in Section 3 for all TruthValue MIB attributes.
* Remove Section 3.6 MIB attributes, and replace with in-line text.
* The attribute name suffixes defined in Section 3 should only be used for TruthValue MIB attributes. Other types of MIB attributes should use different name suffixes.
* Suggest looking at “changes take effect” language, especially “changes take effect at the next MLME-START [or MLME-JOIN]”, this language is suspect. Given our (new) understanding of the lifetime of an instantiation (see Section 2), this likely should be the next MLME-RESET. Or, perhaps the MIB attribute value does change dynamically, in which case START or JOIN may not be the critical points in time where these changes occur, and the correct points of inflection should be identified and described. This common wording is likely just due to cut-and-paste without careful consideration.

1. The “supplier and acquirer, or potential acquirer” in PICS terminology [↑](#footnote-ref-2)
2. The “protocol implementer” in PICS terminology. [↑](#footnote-ref-3)
3. The “user, or potential user” in PICS terminology [↑](#footnote-ref-4)
4. This assumes a correction is made to IEEE Std 802.11-2016 such that all STA types (not just APs) must initialize with MLME-RESET before performing other MAC operations. [↑](#footnote-ref-5)