802.1X YANG Module Specifications

Introduction

The YANG module definitions, presented here, are based on the existing IEEE 802.1X-2010 PAE management information model. This model is represented in Figure 1 below.

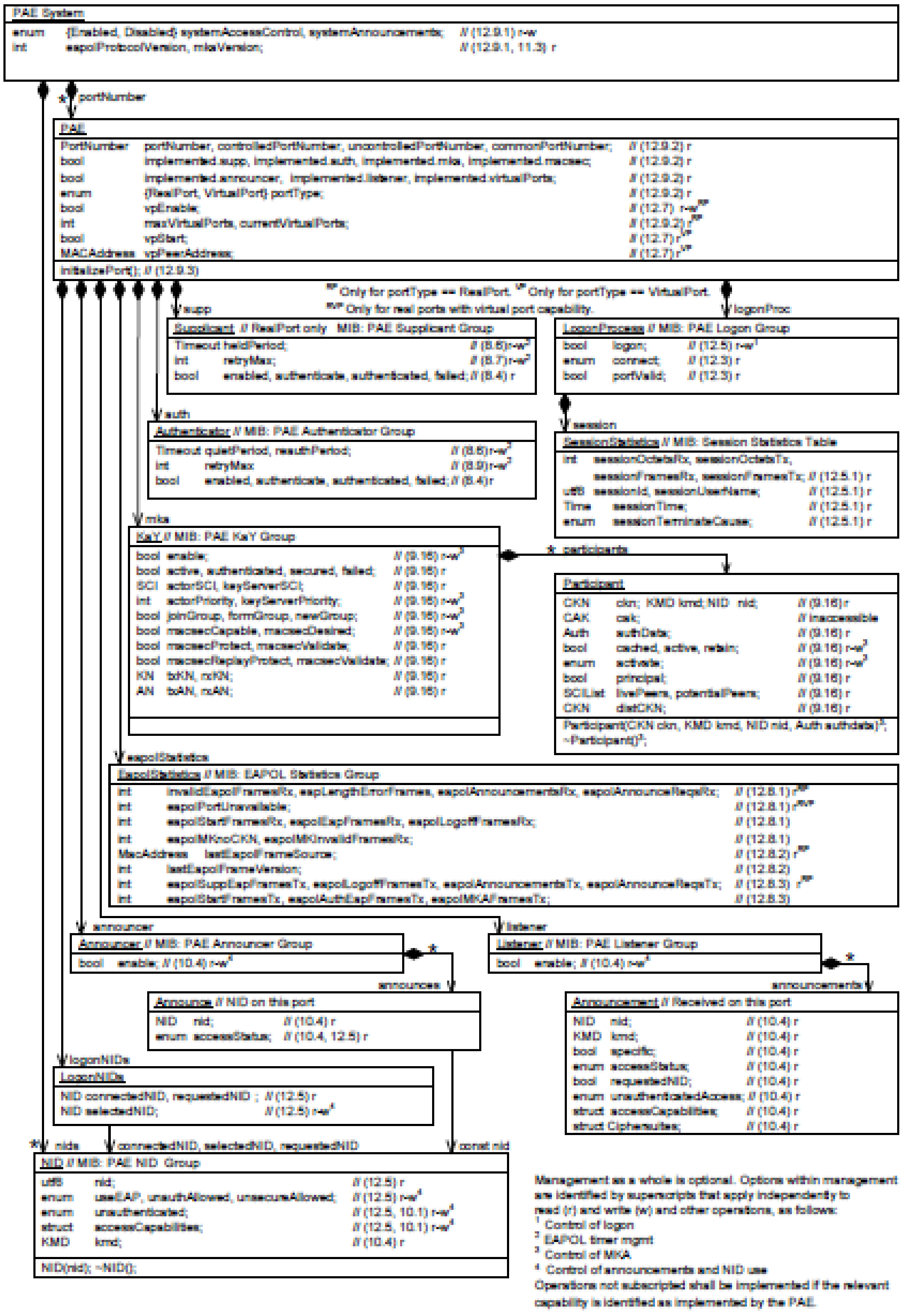
PAE System
enum {Enabled, Disabled} systemAccessControl, systemAnnouncements; // (12.9.1) r-w
int eapolProtocolVersion, mkaVersion; // (12.9.1, 11.3) r
* portNumber
PAE
PortNumber portNumber, controlledPortNumber, uncontrolledPortNumber, commonPortNumber; // (12.9.2) r
bool implemented.supp, implemented.auth, implemented.mka, implemented.macsec; // (12.9.2) r
bool implemented.announcer, implemented.listener, implemented.virtualPorts; // (12.9.2) r
enum {RealPort, VirtualPort} portType; // (12.9.2) r
bool vpEnable; // (12.7) r-w RP
int maxVirtualPorts, currentVirtualPorts; // (12.9.2) r RP
bool vpStart; // (12.7) r VP
MACAddress vpPeerAddress; // (12.7) r VP
initializePort(); // (12.9.3)
RP Only for portType == RealPort. VP Only for portType == VirtualPort.
RVP Only for real ports with virtual port capability.
supp
logonProc
Supplicant // RealPort only MIB: PAE Supplicant Group
Timeout heldPeriod; // (8.6) r-w 2
int retryMax; // (8.7) r-w 2
bool enabled, authenticate, authenticated, failed; // (8.4) r
LogonProcess // MIB: PAE Logon Group
bool logon; // (12.5) r-w 1
enum connect; // (12.3) r
bool portValid; // (12.3) r
auth
session
Authenticator // MIB: PAE Authenticator Group
Timeout quietPeriod, reauthPeriod; // (8.6) r-w 2
int retryMax // (8.9) r-w 2
bool enabled, authenticate, authenticated, failed; // (8.4) r
SessionStatistics // MIB: Session Statistics Table
int sessionOctetsRx, sessionOctetsTx,
sessionFramesRx, sessionFramesTx; // (12.5.1) r
utf8 sessionId, sessionUserName; // (12.5.1) r
Time sessionTime; // (12.5.1) r
enum sessionTerminateCause; // (12.5.1) r
mka
KaY // MIB: PAE KaY Group
bool enable; // (9.16) r-w 3
bool active, authenticated, secured, failed; // (9.16) r
SCI actorSCI, keyServerSCI; // (9.16) r
int actorPriority, keyServerPriority; // (9.16) r-w 3
bool joinGroup, formGroup, newGroup; // (9.16) r-w 3
bool macsecCapable, macsecDesired; // (9.16) r-w 3
bool macsecProtect, macsecValidate; // (9.16) r
bool macsecReplayProtect, macsecValidate; // (9.16) r
KN txKN, rxKN; // (9.16) r
AN txAN, rxAN; // (9.16) r
* participants
Participant
CKN ckn; KMD kmd; NID nid; // (9.16) r
CAK cak; // inaccessible
Auth authData; // (9.16) r
bool cached, active, retain; // (9.16) r-w 3
enum activate; // (9.16) r-w 3
bool principal; // (9.16) r
SCIList livePeers, potentialPeers; // (9.16) r
CKN distCKN; // (9.16) r
Participant(CKN ckn, KMD kmd, NID nid, Auth authdata) 3;
~Participant() 3;
eapolStatistics
EapolStatistics // MIB: EAPOL Statistics Group
int invalidEapolFramesRx, eapLengthErrorFrames, eapolAnnouncementsRx, eapolAnnounceReqsRx; // (12.8.1) r RP
int eapolPortUnavailable; // (12.8.1) r RVP
int eapolStartFramesRx, eapolEapFramesRx, eapolLogoffFramesRx; // (12.8.1)
int eapolMKnoCKN, eapolMKInvalidFramesRx; // (12.8.1)
MacAddress lastEapolFrameSource; // (12.8.2) r RP
int lastEapolFrameVersion; // (12.8.2)
int eapolSuppEapFramesTx, eapolLogoffFramesTx, eapolAnnouncementsTx, eapolAnnounceReqsTx; // (12.8.3) r RP
int eapolStartFramesTx, eapolAuthEapFramesTx, eapolMKAFramesTx; // (12.8.3)
announcer
listener
Announcer // MIB: PAE Announcer Group
bool enable; // (10.4) r-w 4
Listener // MIB: PAE Listener Group
bool enable; // (10.4) r-w 4
*
announces
*
announcements
Announce // NID on this port
NID nid; // (10.4) r
enum accessStatus; // (10.4, 12.5) r
Announcement // Received on this port
NID nid; // (10.4) r
KMD kmd; // (10.4) r
bool specific; // (10.4) r
enum accessStatus; // (10.4) r
bool requestedNID; // (10.4) r
enum unauthenticatedAccess; // (10.4) r
struct accessCapabilities; // (10.4) r
struct Ciphersuites; // (10.4) r
logonNIDs
LogonNIDs
NID connectedNID, requestedNID ; // (12.5) r
NID selectedNID; // (12.5) r-w 1
* nids
connectedNID, selectedNID, requestedNID
const nid
NID // MIB: PAE NID Group
utf8 nid; // (12.5) r
enum useEAP, unauthAllowed, unsecureAllowed; // (12.5) r-w 4
enum unauthenticated; // (12.5, 10.1) r-w 4
struct accessCapabilities; // (12.5, 10.1) r-w 4
KMD kmd; // (10.4) r
NID(nid); ~NID();
Management as a whole is optional. Options within management are identified by superscripts that apply independently to read (r) and write (w) and other operations, as follows:
1 Control of logon
2 EAPOL timer mgmt
3 Control of MKA
4 Control of announcements and NID use
Operations not subscripted shall be implemented if the relevant capability is identified as implemented by the PAE.

Figure 1: 802.1X-2010 PAE Information Model

The YANG modules will provide configuration (and notification) management for 802.1X specified port-based network access controls. Port-based network access control allows a network administrator to restrict the use of IEEE 802 LAN service access points (ports) to secure communication between authenticated and authorized devices. IEEE Std 802.1X specifies an architecture, functional elements, and protocols that support mutual authentication between the clients of ports attached to the same LAN and secure communication between the ports.

The control allows a port to be reinitialized, terminating (and potentially restarting) authentication exchanges and MKA operation, based on a data model described in a set of YANG modules.

This (proposed) amendment (??) to IEEE-802.1X will specify the YANG data models that provide configuration management for IEEE 802.1X port-based network access controls.

802.1X and Netconf/YANG

Netconf is a widely accepted configuration management protocol that promises to simplify network configuration. YANG is a formalized data modeling language used to model configuration and state data that can be used by Netconf. The adoption of YANG, will allow IEEE 802.1 Bridging vendors (that provide port-based network access controls) as well as Network Management Systems to speak a common language, and thus simplify Service Provider operations.

Service Providers are moving towards a Netconf/YANG configuration management paradigm. As such vendors that provide IEEE 802.1 bridges and supporting functionality would be interested in this amendment.

Currently, other SDOs (e.g., MEF and IETF) are developing YANG data models. IEEE 802 needs to provide specifications of YANG data models in support of IEE 802.1 managed entities. YANG data model have already been defined by other SDOs (e.g., MEF and IETF). For example, MEF have defined YANG models for Service OAM Fault Management & Performance Monitoring, and IETF have defined YANG data models for Interface Management.

YANG Module Definitions

The (preliminary/draft) YANG modules in support of the PAE management information model are provided below. This may change during the drafting process.

NOTE: The current YANG module definitions defines an 802.1X configuration that refers to a list of ports that may want to employee the 802.1X configuration management.

Alternatively, a scheme where the port that is identified by the (network) operator, is augmented with 802.1X configuration, could also be provided. This is a typical scheme to support (port based) YANG models today by (system) vendors.

The preliminary/draft YANG module definition is shown below:

```
module dot1x {

  namespace "ieee:ns:yang:ieee-port-dot1x";
  prefix "dot1x";

  import ieee-types { prefix ieee; }

  organization
    "Institute of Electrical and Electronics Engineers";

  contact
    "Web URL: http://ieee.org/
      E-mail:  corporate-communications@ieee.org
      Postal:
           U.S.A.
      Phone:   +1 732-563-6820
      Fax:     +1 732-981-9511";

        description
    "Port-based network access control allows a network administrator to restrict the use of IEEE 802 LAN
                service access points (ports) to secure communication between authenticated and authorized devices. IEEE
                Std 802.1X specifies an architecture, functional elements, and protocols that support mutual authentication
                between the clients of ports attached to the same LAN and secure communication between the ports.

                The following control allows a port to be reinitialized, terminating (and potentially restarting)
                authentication exchanges and MKA operation, based on a data model described in a set of YANG
modules.";

  revision "2015-03-10" {
    description
      "Initial Version.";
    reference
      "IEEE 802.1X-2010, PAE (Port Access Entity) management information model.";
  }

        // Type definitions used by dot1X YANG module

        typedef nidType {
                type string {
                        // Need to confirm this.
                        length "0..100";
                }
        }

        typedef sessionUserNameType {
                description "utf8 definition";
                type string {
                        length "0..253";
                }
        }

        typedef sessionIdType {
                description "utf8 definition";
                type string {
                        length "3..253";
                }
        }

        typedef accessCapabilitiesType {
                type bits {
                        bit eap {
```

```
                    position 0;
                }
                bit mka {
                    position 1;
                }
                bit macsec {
                    position 2;
                }
                bit higher-layer {
                    position 3;
                }
                bit higher-layer-fallback {
                    position 4;
                }
                bit vendor-specific {
                    position 5;
                }
            }
        }

        typedef portNumberType {
            type uint16;
        }

        typedef keyNumberType {
            type uint32;
        }

        typedef associationNumberType {
            type uint16;
        }

        typedef sciListType {
            // To be defined.
            type uint16;
        }

        typedef cknType {
            // To be defined.
            type uint16;
        }

        typedef kmdType {
            // To be defined.
            type uint16;
        }

        typedef authType {
            // To be defined.
            type uint16;
        }

        typedef cipherSuitesType {
            // Structure to be defined.
            type uint16;
        }

        grouping nidGroup {
```

```
list pae-nid-group {
        key "nids";
        leaf nids {
                type sessionUserNameType;
        }
        leaf use-eap {
                type enumeration {
                        enum never;
                        enum immediate;
                        enum mka-fail;
                }
                description
                        "Determines when the Logon Process will initiate EAP, if the Supplicant and or
                         Authenticator are enabled.";
        }
        leaf unauth-allowed {
                type enumeration {
                        enum never;
                        enum immediate;
                        enum auth-fail;
                }
                description
                        "Determines when the Logon Process will tell the CP state machine to provide
                         unauthenticated connectivity.";
        }
        leaf unsecure-allowed {
                type enumeration {
                        enum never;
                        enum immediate;
                        enum mka-fail;
                        enum mka-server;
                }
                description
                        "Determines when the Logon Process will tell the CP state machine to provide
                         authenticated but unsecured connectivity.";
        }
        leaf unauthenticated-access {
                type enumeration {
                        enum no-access;
                        enum fallback-access;
                        enum limited-access;
                        enum open-access;
                }
        }
        leaf access-capabilities {
                type accessCapabilitiesType;
                description "Authentication mechanisms.";
        }
        container nid-status {
                config false;
                leaf kmd {
                        type kmdType;
                }
                leaf nid {
                        type nidType;
                }
        }
}
```

```
        }

        // Configuration objects used by 802.1X YANG module

        // PAE object
        container PAE-system {
                leaf systemAccessControl {
                        mandatory true;
                        type enumeration {
                                enum disabled;
                                enum enabled;
                        }
                        description
                                "Setting this control to 'disabled' deletes any virtual ports previously instantiated,
                                 and terminates authentication exchanges and MKA operation. Each real port PAE
                                 behaves as if enabledVirtualPorts was clear, the PAEs Supplicant, Authenticator,
                                 and KaY as if their enabled controls were clear, and Logon Process(es) as if
unauthAllowed
                                 was Immediate. Announcements can be transmitted (subject to other controls), both
periodically
                                 and in response to announcement requests (conveyed by EAPOL-Starts or EAPOL-
Announcement-Reqs)
                                 but are sent with a single NID Set, with a null NID, and the Access Information TLV (and
no other)
                                 with an accessStatus of No Access, accessRequested false, OpenAccess, and no
accessCapabilities.
                                 The control variable settings for each real port PAE are unaffected, and will be used
once
                                 systemAccessControl is set to 'enabled'";
                }
                leaf systemAnnouncements {
                        mandatory true;
                        type enumeration {
                                enum disabled;
                                enum enabled;
                        }
                        description
                                "Setting this control to Disabled causes each PAE to behave as if enabled were clear for
                                 the PAE's Announcement functionality. The independent controls for each PAE apply
                                 if systemAnnouncements is Enabled.";
                }
                container status {
                        config false;
                        leaf eapolProtocolVersion {
                                type uint32;
                                description "The EAPOL protocol version for this system.";
                        }
                        leaf mkaVersion {
                                type uint32;
                                description "The MKA protocol version for this system.";
                        }
                }

                // Port Authentication Entity (PAE) Object
                list PAE {
                        key "port-number";
                        leaf port-number {
                                type portNumberType;
```

```
}
leaf port-type {
        description "The port type of the PAE.";
        config false;
        type enumeration {
                enum realPort;
                enum virtualPort;
        }
}
leaf virtual-port-enabled {
        type boolean;
        description
                "A real port's PAE may be configured to create virtual ports
                 to support mlti-access LANs provided that MKA and MACsec
                 operation is enabled for that port.";
}
leaf controlled {
        config false;
        type portNumberType;
}
leaf uncontrolled {
        config false;
        type portNumberType;
}
leaf common {
        config false;
        type portNumberType;
}
container implemented {
        config false;
        leaf supp {
                type boolean;
        }
        leaf auth {
                type boolean;
        }
        leaf mka {
                type boolean;
        }
        leaf macsec {
                type boolean;
        }
        leaf announcer {
                type boolean;
        }
        leaf listener {
                type boolean;
        }
        leaf virtual-ports {
                type boolean;
        }
}
container virtual-port {
        config false;
        leaf max {
                type uint16;
        }
        leaf current {
```

```
                        type uint16;
                }
                leaf start {
                        type uint16;
                }
                leaf peer-address {
                        type ieee:mac-address;
                }
        }

        // SUPPLICANT Object
        list supplicant {
                key "supp";
                leaf supp {
                        type portNumberType;
                }
                leaf held-period {
                        type uint16;
                        units seconds;
                }
                leaf retry-max {
                        type uint8;
                        description
                                "Specifies the maximum number of re-authentication attempts on
an
                                 authenticator port before port is unauthorized.";
                }
                container status {
                        config false;
                        leaf enabled {
                                type boolean;
                        }
                        leaf authenticate {
                                type boolean;
                        }
                        leaf authenticated {
                                type boolean;
                        }
                        leaf failed {
                                type boolean;
                        }
                }
        }

        // AUTHENTICATOR Object
        list authenticator {
                key "auth";
                leaf auth {
                        type uint8;
                }
                leaf quiet-period {
                        type uint16;
                        units seconds;
                        default "60";
                        description
                                "Number of seconds that the switch remains in the
                                 quiet state following a failed authentication exchange with
                                 the client";
```

```
                                        reference
                                                "IEEE 802.1X Clause 8.6, Figure 12-3";
                                }
                                leaf reauth-period {
                                        type uint16;
                                        units seconds;
                                        default "3600";
                                        description
                                                "This object indicates the time period of the
                                                 reauthentication to the supplicant.";
                                        reference
                                                "IEEE 802.1X Clause 8.6, Figure 12-3";
                                }
                                leaf retry-max {
                                        type uint8;
                                        description
                                                "Specifies the maximum number of re-authentication attempts on
an
                                                 authenticator port before port is unauthorized.";
                                }
                                container status {
                                        config false;
                                        leaf enabled {
                                                type boolean;
                                        }
                                        leaf authenticate {
                                                type boolean;
                                        }
                                        leaf authenticated {
                                                type boolean;
                                        }
                                        leaf failed {
                                                type boolean;
                                        }
                                }
                        }

                        // PAE KaY (Group) Object
                        list kay {
                                key "mka";
                                leaf mka {
                                        type uint8;
                                }
                                leaf enable {
                                        type boolean;
                                }
                                container actor {
                                        leaf priority {
                                                type uint8;
                                        }
                                        container sci {
                                                config false;
                                                leaf mac-address {
                                                        type ieee:mac-address;
                                                }
                                                leaf port-id {
                                                        type portNumberType;
                                                }
```

```
        }
    }
    container key-server {
        leaf priority {
            type uint8;
        }
        container sci {
            config false;
            leaf mac-address {
                type ieee:mac-address;
            }
            leaf port-id {
                type portNumberType;
            }
        }
    }
    container group {
        leaf join {
            type boolean;
        }
        leaf form {
            type boolean;
        }
        leaf new {
            type boolean;
        }
    }
    container macsec {
        leaf capable {
            type boolean;
        }
        leaf desired {
            type boolean;
        }
        leaf protect {
            config false;
            type boolean;
        }
        leaf validate {
            config false;
            type boolean;
        }
        leaf replay-protect {
            config false;
            type boolean;
        }
    }
    container status {
        config false;
        leaf active {
            type boolean;
        }
        leaf authenticated {
            type boolean;
        }
        leaf secured {
            type boolean;
        }
```

```
		leaf failed {
			type boolean;
		}
	}
	container key-stats {
		config false;
		leaf txKN {
			type keyNumberType;
		}
		leaf rxKN {
			type keyNumberType;
		}
	}
	container association-stats {
		config false;
		leaf txAN {
			type associationNumberType;
		}
		leaf rxAN {
			type associationNumberType;
		}
	}

	// PARTICIPANT Object

	list participant {
		key "participants";
		leaf participants {
			type uint16;
		}
		leaf cached {
			type boolean;
		}
		leaf active {
			type boolean;
		}
		leaf retain {
			type boolean;
		}
		leaf activate {
			type enumeration {
				enum principal;
				enum livePeers;
				enum potentialPeers;
				enum distCKN;
			}
		}
		container peers {
			config false;
			leaf live {
				type sciListType;
			}
			leaf potential {
				type sciListType;
			}
		}
		leaf ckn {
			config false;
```

```
                                        type cknType;
                                }
                                leaf kmd {
                                        config false;
                                        type kmdType;
                                }
                                leaf nid {
                                        type nidType;
                                }
                                leaf authData {
                                        config false;
                                        type authType;
                                }
                                leaf principal {
                                        type boolean;
                                }
                                leaf distCKN {
                                        config false;
                                        type cknType;
                                }
                        }
                }

                // LOGIN-NID object
                list logon-nid {
                        key "nids";
                        leaf nids {
                                type nidType;
                        }
                        leaf selected-nid {
                                type nidType;
                        }
                        container nid-type {
                                config false;
                                leaf connected {
                                        type nidType;
                                }
                                leaf requested {
                                        type nidType;
                                }
                        }
                        uses nidGroup;
                }

                // ANNOUNCER Object
                list announcer {
                        key "announcers";
                        leaf announcers {
                                type uint16;
                        }
                        leaf enable {
                                type boolean;
                        }
                        list announce {
                                key "announces";
                                leaf announces {
                                        type uint16;
```

```
                    }

                    leaf nid {
                        config false;
                        type nidType;
                    }
                    leaf access-status {
                        config false;
                        // Need to confirm these types.
                        type enumeration {
                            enum no-access;
                            enum acessRequested-fail;
                            enum openAccess;
                            enum no-accessCapabilities;
                        }
                    }
                }
                uses nidGroup;
            }

            // LISTENER Object
            list listener {
                key "listeners";
                leaf listeners {
                    type uint8;
                }
                leaf enable {
                    type boolean;
                }
                list announcement {
                    key "announcements";
                    leaf announcements {
                        type uint8;
                    }

                    leaf nid {
                        config false;
                        type nidType;
                    }
                    leaf kmd {
                        config false;
                        type kmdType;
                    }
                    leaf specific {
                        config false;
                        type boolean;
                    }
                    leaf access-status {
                        config false;
                        // Need to confirm these types.
                        type enumeration {
                            enum no-access;
                            enum acessRequested-fail;
                            enum openAccess;
                            enum no-accessCapabilities;
                        }
                    }
                    leaf requested-nid {
```

```
                                        config false;
                                        type boolean;
                                    }
                                    leaf unauthenticated-access {
                                        config false;
                                        // Need to confirm these types.
                                        type enumeration {
                                            enum no-access;
                                            enum acessRequested-fail;
                                            enum openAccess;
                                            enum no-accessCapabilities;
                                        }
                                    }
                                    leaf access-capabilities {
                                        config false;
                                        type accessCapabilitiesType;
                                    }
                                    leaf cipher-suites {
                                        config false;
                                        type cipherSuitesType;
                                    }
                                }
                            }
                        }
                        uses nidGroup;
                    }
}
```

802.1X YANG Module Tree View

The tree view is illustrated below.

```
module: dot1x
  +-rw PAE-system
    +-rw systemAccessControl    enumeration
    +-rw systemAnnouncements    enumeration
    +-ro status
    |  +-ro eapolProtocolVersion?   uint32
    |  +-ro mkaVersion?             uint32
    +-rw PAE* [port-number]
    |  +-rw port-number             portNumberType
    |  +-ro port-type?              enumeration
    |  +-rw virtual-port-enabled?   boolean
    |  +-ro controlled?             portNumberType
    |  +-ro uncontrolled?           portNumberType
    |  +-ro common?                 portNumberType
    |  +-ro implemented
    |  |  +-ro supp?            boolean
    |  |  +-ro auth?            boolean
    |  |  +-ro mka?             boolean
    |  |  +-ro macsec?          boolean
    |  |  +-ro announcer?       boolean
    |  |  +-ro listener?        boolean
    |  |  +-ro virtual-ports?   boolean
    |  +-ro virtual-port
    |  |  +-ro max?         uint16
    |  |  +-ro current?     uint16
    |  |  +-ro start?       uint16
```

```
 |  |  +-ro peer-address?   ieee:mac-address
 |  +-rw supplicant* [supp]
 |  |  +-rw supp           portNumberType
 |  |  +-rw held-period?   uint16
 |  |  +-rw retry-max?     uint8
 |  |  +-ro status
 |  |     +-ro enabled?         boolean
 |  |     +-ro authenticate?    boolean
 |  |     +-ro authenticated?   boolean
 |  |     +-ro failed?          boolean
 |  +-rw authenticator* [auth]
 |  |  +-rw auth            uint8
 |  |  +-rw quiet-period?    uint16
 |  |  +-rw reauth-period?   uint16
 |  |  +-rw retry-max?       uint8
 |  |  +-ro status
 |  |     +-ro enabled?         boolean
 |  |     +-ro authenticate?    boolean
 |  |     +-ro authenticated?   boolean
 |  |     +-ro failed?          boolean
 |  +-rw kay* [mka]
 |  |  +-rw mka                 uint8
 |  |  +-rw enable?             boolean
 |  |  +-rw actor
 |  |  |  +-rw priority?   uint8
 |  |  |  +-ro sci
 |  |  |     +-ro mac-address?   ieee:mac-address
 |  |  |     +-ro port-id?       portNumberType
 |  |  +-rw key-server
 |  |  |  +-rw priority?   uint8
 |  |  |  +-ro sci
 |  |  |     +-ro mac-address?   ieee:mac-address
 |  |  |     +-ro port-id?       portNumberType
 |  |  +-rw group
 |  |  |  +-rw join?   boolean
 |  |  |  +-rw form?   boolean
 |  |  |  +-rw new?    boolean
 |  |  +-rw macsec
 |  |  |  +-rw capable?         boolean
 |  |  |  +-rw desired?         boolean
 |  |  |  +-ro protect?         boolean
 |  |  |  +-ro validate?        boolean
 |  |  |  +-ro replay-protect?   boolean
 |  |  +-ro status
 |  |  |  +-ro active?         boolean
 |  |  |  +-ro authenticated?   boolean
 |  |  |  +-ro secured?        boolean
 |  |  |  +-ro failed?         boolean
 |  |  +-ro key-stats
 |  |  |  +-ro txKN?   keyNumberType
 |  |  |  +-ro rxKN?   keyNumberType
 |  |  +-ro association-stats
 |  |  |  +-ro txAN?   associationNumberType
 |  |  |  +-ro rxAN?   associationNumberType
 |  |  +-rw participant* [participants]
 |  |     +-rw participants    uint16
 |  |     +-rw cached?         boolean
 |  |     +-rw active?         boolean
```

```
| |    +-rw retain?        boolean
| |    +-rw activate?      enumeration
| |    +-ro peers
| |    | +-ro live?       sciListType
| |    | +-ro potential?   sciListType
| |    +-ro ckn?           cknType
| |    +-ro kmd?           kmdType
| |    +-rw nid?           nidType
| |    +-ro authData?      authType
| |    +-rw principal?     boolean
| |    +-ro distCKN?       cknType
| +-rw logon-nid* [nids]
| | +-rw nids            nidType
| | +-rw selected-nid?   nidType
| | +-ro nid-type
| | | +-ro connected?   nidType
| | | +-ro requested?   nidType
| | +-rw pae-nid-group* [nids]
| |    +-rw nids                    sessionUserNameType
| |    +-rw use-eap?                enumeration
| |    +-rw unauth-allowed?         enumeration
| |    +-rw unsecure-allowed?       enumeration
| |    +-rw unauthenticated-access?   enumeration
| |    +-rw access-capabilities?     accessCapabilitiesType
| |    +-ro nid-status
| |       +-ro kmd?   kmdType
| |       +-ro nid?   nidType
| +-rw announcer* [announcers]
| | +-rw announcers       uint16
| | +-rw enable?          boolean
| | +-rw announce* [announces]
| | | +-rw announces       uint16
| | | +-ro nid?            nidType
| | | +-ro access-status?   enumeration
| | +-rw pae-nid-group* [nids]
| |    +-rw nids                    sessionUserNameType
| |    +-rw use-eap?                enumeration
| |    +-rw unauth-allowed?         enumeration
| |    +-rw unsecure-allowed?       enumeration
| |    +-rw unauthenticated-access?   enumeration
| |    +-rw access-capabilities?     accessCapabilitiesType
| |    +-ro nid-status
| |       +-ro kmd?   kmdType
| |       +-ro nid?   nidType
| +-rw listener* [listeners]
|    +-rw listeners       uint8
|    +-rw enable?         boolean
|    +-rw announcement* [announcements]
|       +-rw announcements            uint8
|       +-ro nid?                     nidType
|       +-ro kmd?                     kmdType
|       +-ro specific?                boolean
|       +-ro access-status?           enumeration
|       +-ro requested-nid?           boolean
|       +-ro unauthenticated-access?   enumeration
|       +-ro access-capabilities?     accessCapabilitiesType
|       +-ro cipher-suites?           cipherSuitesType
+-rw pae-nid-group* [nids]
```

```
+-rw nids                    sessionUserNameType
+-rw use-eap?                enumeration
+-rw unauth-allowed?         enumeration
+-rw unsecure-allowed?       enumeration
+-rw unauthenticated-access? enumeration
+-rw access-capabilities?    accessCapabilitiesType
+-ro nid-status
  +-ro kmd?   kmdType
  +-ro nid?   nidType
```

Outstanding Specification Areas

The following specification areas need to be resolved/discussed, moving forward:

1. YANG configuration model based on an augmentation of the (existing) YANG port model (e.g., via ietf-interfaces.yang) or based upon the identification of a port that may participate in 802.1X configuration.
2. YANG 802.1X type definitions associated with NID type, SCI list type, CKN type, KMD type, Auth type, cipher suites type, etc.
3. General structure of the YANG model.
4. Deviation from current MIB structure (or PAE information model), to support the YANG configuration model.

To be completed.