IEEE P802.1CBcv
Usage of AutonomousType in the MIB

Stephan Kehrer, Hirschmann Automation and Control GmbH

September 2020
Motivation

- Several comments on the MIBs presented in draft D0.4 of IEEE P802.1CBcv asked for clarification on the usage of *AutonomousType* in the MIB

- This presentation is intended to provide some hints and clarifications on the construct of *AutonomousType* and its usage
What is AutonomousType?

- **AutonomousType** is a textual convention defined in [RFC 2579](https://tools.ietf.org/html/rfc2579)

- The description in the RFC is:
  
  *Represents an independently extensible type identification value. It may, for example, indicate a particular sub-tree with further MIB definitions, or define a particular type of protocol or hardware.*

- It allows to use sub-tables (i.e. tables that are conceptually part of another table) by providing a means to reference them by “pointer”
IEEE Std 802.3CB contains several tables that
- require flexibility in the amount of entries they can contain
- require flexibility in the managed objects that are part of the sub-tree but are included in another set of managed objects that are modeled as a table in the MIB

Examples are:
- the stream identity table, clause 9.1
- the sequence generation table, clause 10.3
- ...
The *tsnStreamIdentification* type defines the stream identification function used to identify a specific stream.

Depending on the stream identification function a dedicated subset of parameters is required (i.e. Null Stream identification requires different objects than IP Stream identification).

This is modeled in the MIB using *AutonomousType* as shown on the next slides.

The usage of *AutonomousType* to model the stream identification type is modeled the same way as *hrDeviceType* is modeled in RFC 2790.
Example `tsnStreamIdentificationType` (clause 9.1.1.6)

- Definition of the `StreamIdentificationTypes`
- Definition of an object `StreamIdentificationTypeSelect` in the stream identification table
  - is of type `AutonomousType`
  - contains one of the previously defined `StreamIdentificationTypes`
  - description clarifies which sub-table is used for the respective identification types
- Definition of the sub-tables with the required set of managed objects for the respective stream identification methods
- Sub-table entry is linked to the parent table by using the index of the parent entry as its own index
Example `tsnStreamIdentificationType` (clause 9.1.1.6)

- Definition of the `StreamIdentificationTypes`
- Definition of an object `StreamIdentificationTypeSelect` in the stream identification table
Definition of the sub-tables with the required set of managed objects for that stream identification

Sub-table entry is linked to the parent table by using the index of the parent entry as its own index
Example *tsnStreamIdInFacOutputPortList* (clause 9.1.1.2)

- *tsnStreamIdInFacOutputPortList* contains a list of ports “[…] on which an in-facing Stream identification function using this identification method is to be placed for this Stream in the output (towards the system forwarding function) direction[…]”

- This is modeled in the MIB using *AutonomousType* as shown on the next slides
  - The length of the list is unknown, so other ways to model this (e.g. OCTET STRING) did not seem appropriate or unnecessarily restraining
  - The 802.1 bridge MIB defines a type *PortList*. Can’t we use that instead?
    - *PortList* is effectively a bitmap representing the physical ports existing in a bridge
    - In IEEE Std 802.1CB the usage of port does not refer to physical bridge ports, it refers to interfaces
  → For this reason we can not use *PortList*; the entries in the port lists modeled in the MIB need to be references to interfaces, not to physical bridge ports
Example tsnStreamIdInFacOutputPortList (clause 9.1.1.2)

- Definition of an object ieee8021StreamIdStreamIdInFacOutputPortList in the stream identification table
  - is of type AutonomousType
  - contains OID of the sub-table that actually holds the port/interface entries
  - description clarifies which sub-table is used for the respective identification types

- Definition of the sub-table with the required set of managed objects for the port list

- Sub-table entry is linked to the parent table by using the index of the parent entry as its own index
Example *tnsStreamIdInFacOutputPortList* (clause 9.1.1.2)

- Definition of an object

  *StreamIdentificationTypeSelect* in

  the stream identification table
Definition of the sub-table with the required set of managed objects for the port list

- Sub-table entry is linked to the parent table by using the index of the parent entry as its own index
  - Additional index is required because each entry in the sub-table contains exactly one interface
  - A list of interfaces in an entry of the parent table requires several entries in the sub-table
Any questions?