**5.4 Stream identifictaion optional behaviors**

In addition to the requirements of 5.2, a system performing Stream identifcation may optionally be able to

perform the following functions:

a)The items in 5.2 and 5.3 on more than one port;

b) The items in 5.2 and 5.3 for some number of Compound Streams greater than 1;

c) An IP octuple Stream identification function (6.7);

d) An EtherType Stream identification function (6.8);and/or

e) Additional types of Stream Identification functions.

**5.7 Talker end system optional behaviors**

In addition to the requirements of 5.5, a Talker end system may optionally be able to perform the following

functions:

a) The items in 5.5 and 5.6 on more than one port;

b) The items in 5.5 fand 5.6 or some number of Compound Streams greater than 1;

c) An IP octuple Stream identification function (6.7);

d) An EtherType Stream identification function (6.8);

e) Additional types of Stream Identification functions;

f) The HSR sequence tag (7.9);

g) The PRP sequence trailer (7.10); and/or

h) Additional types of Sequence encode/decode functions.

**5.10 Listener end system optional behaviors**

In addition to the requirements of 5.8, a Listener end system may optionally be able to perform the following

functions:

a) The items in 5.8 and 5.9 on more than one port;

b) The items in 5.8 and 5.9 for some number of Compound Streams greater than 1;

c) An IP octuple Stream identification function (6.7);

d) An EtherType Stream identification function (6.8);

e) Additional types of Stream Identification function functions;

f) The HSR sequence tag (7.9);

g) The PRP sequence trailer (7.10);

h) Additional types of Sequence encode/decode functions; and/or

i) At least two instances of Individual recovery functions (7.5), each using the VectorRecoveryAlgorithm (7.4.3.4).

**5.12 Relay system recommended behaviors**

A relay system should be able to instantiate the following in-facing functions on at least two ports, for both

transmit and receive, for at least one Stream:

a) Active Destination MAC and VLAN Stream identification functions (6.6) for encoding and

decoding packets; and

b) IP octuple Stream identification functions (6.7) for identifying packets; or

c) EtherType Stream identification functions (6.8)

**6. Stream identification**

….

***[line 31]***

NOTE—In principle, any number of different methods for identifying and encoding Streams can be defined. Several required methods are specified in the following sections (6.4, 6.5, 6.6, 6.7, 6.8).

***[line 38]***

Stream identification is described in the following subclauses as follows:

a) Additional service subparameters required by Stream identification are in 6.1.

b) The Stream Identification function is described in 6.2, and its placement in the protocol stack of a

system in 6.3.

c) Five specific Stream Identification functions are described: Null Stream identification (6.4), Source

MAC and VLAN Stream identification (6.5), Active Destination MAC and VLAN Stream

identification (6.6), IP octuple Stream identification (6.7), and EtherType Stream identification (6.8).

These Stream Identification function are summarized in Table 6-1.

**Table 6-1—Stream Identification functions**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Stream Identification function** | **active/passive** | **examines** | **overwrites** | **See** |
| Null Stream identification | passive | destination\_address, vlan\_identifier | none | 6.4, 9.1.2 |
| Source MAC and VLAN Stream identification | passive | source\_address, vlan\_identifier | none | 6.5, 9.1.3 |
| Active Destination MAC and VLAN Stream identification  | active | destination\_address, vlan\_identifier | destination\_address, vlan\_identifier, priority | 6.6, 9.1.4 |
| IP octuple Stream identification | passive | destination\_address, vlan\_identifier, IP source address, IP destination address, IP next protocol, source port, destination port | none | 6.7, 9.1.5 |
| EtherType Stream identification | passive | destination\_address, EtherType, vlan\_identifier | none | 6.8, 9.1.6 |

**6.6 Active Destination MAC and VLAN Stream identification**

….

***[line 51]***

NOTE 2—Changing the destination MAC address and/or VLAN must be done carefully, if the receiver is to recognize the packet. For example, if Active Destination MAC and VLAN Stream identification is used along with IP octuple Stream identification (6.7) or EtherType Stream identification (6.8), the user can configure Active Destination MAC and VLAN Stream identification at the receiving end to restore the original destination MAC address and VLAN before delivery up the protocol stack.

**6.8 EtherType Stream identification**

The EtherType Stream identification function is a passive Stream Identification function that operates at the frame level. It can be defined using the Enhanced Internal Sublayer Service (EISS) described in subclause 6.9 of IEEE 802.1Q-2014, in which case it is enhanced with the extra stream\_handle subparameter of the connection\_identifier, specified in 6.1 of the present standard. It discards the stream\_handle subparameter for packets passed down the stack. It generates a stream\_handle subparameter on frames passed up the stack based on the frame’s EtherType and EISS parameters. It does not change any of a packet’s parameters. In order to instantiate the EtherType Stream identification function, the tsnStreamIdIdentificationType managed object (9.1.1.6) is encoded using the OUI (00-80-C2) and the type values as shown in Table 9-1.

EtherType Stream identification can be coupled, for example, with Active Destination MAC and VLAN Stream identification (6.6) to assign a particular {MAC address, VLAN, priority} triple to frames transporting a particular protocol.

The managed objects for EtherType identification are described in 9.1.6.

NOTE—The drop\_eligible parameter is also present, along with the VLAN identifier and priority, in an IEEE 802.1Q VLAN tag. FRER does not affect the use of this parameter. It passes through EtherType Stream identification unchanged, and defaults to False when not present.

**8.2 IEEE 802.1Q Bridge input transformations**

The Input transformations, marked with white boxed with boldface type in Figure 8-1, enable a Bridge to proxy for a non-FRER-capable end system. The expanded input port identifies packets belonging to a Stream (e.g., using IP octuple Stream identification, 6.7), serializes the packets with a Sequence generation function (7.4.1), encodes the sequence number with an R-TAG (7.8), and then gives the packets belonging to this Stream a {vlan\_identifier, destination\_mac\_address} pair that is unique, at least inside this Bridge, using Active Destination MAC and VLAN Stream identification (6.6). The IEEE 802.1Q Forwarding Process, enhanced with the Individual recovery function (7.5) and Sequence recovery function (7.4.2), then forwards the frame.

**9.1.1.6 tsnStreamIdIdentificationType**

An enumerated value indicating the method used to identify packets belonging to the Stream. The enumeration includes an Organizationally Unique Identifier (OUI) or Company Identifier (CID) to identify the organization defining the enumerated type. The values defined by this standard are shown in Table 9-1.

**Table 9-1—Stream Identification types**

|  |  |  |  |
| --- | --- | --- | --- |
| **OUI/CID** | **Type number** | **Stream Identification function** | **Controlling parameters** |
| 00-80-C2 | 0 | Reserved |  |
| 00-80-C2 | 1 | Null Stream identification (6.4) | 9.1.2 |
| 00-80-C2 | 2 | Source MAC and VLAN Stream identification (6.5) | 9.1.3 |
| 00-80-C2 | 3 | Active Destination MAC and VLAN Stream identification (6.6) | 9.1.4 |
| 00-80-C2 | 4 | IP octuple Stream identification (6.7) | 9.1.5 |
| 00-80-C2 | 5 | EtherType Stream identification (6.8) | 9.1.6 |
| 00-80-C2 | 6-255 | Reserved |  |
| other |  | Defined by entity owning the OUI or CID |  |

**9.1.6 Managed objects for EtherType Stream identification**

When instantiating an instance of the EtherType Stream identification function (6.8), the parameters in the following subsections replace the tsnStreamIdParameters managed object (9.1.1.7).

**9.1.6.1 tsnCpeEtherTypeDestMac**

Specifies the destination\_address parameter that identifies a packet in an EISS indication primitive.

**9.1.6.2 tsnCpeEtherTypeTagged**

An enumerated value indicating whether a packet in an EISS indication primitive to the EtherType Stream identification function is to have a VLAN tag. It can take the following values:

1) **tagged:** An input frame must have a VLAN tag to be recognized as belonging to the Stream.

An output frame receives a VLAN tag.

2) **priority:** An input frame must be untagged, or have a VLAN tag with a VLAN ID = 0 to be

recognized as belonging to the Stream. An output frame is marked with a VLAN tag with

VLAN ID = 0.

3) **all:** A frame is recognized as belonging to the Stream whether tagged or not. An output frame is

to be untagged.

**9.1.6.3 tsnCpeEtherTypeVlan**

Specifies the vlan\_identifier parameter that identifies a packet in an EISS indication primitive to the EtherType Stream identification function. A value of 0 indicates that the frame is not to have a VLAN tag.

**9.1.6.4 tsnCpeEtherTypeEtherType**

Specifies the value that must be matched by the first two octets of the mac\_service\_data\_unit to identify packets coming up from lower layers

**Annex A**

**A.2.2 Stream identification system**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Item** | **Feature** | **Subclause** | **Value/Comment** | **Status** | **Support** |
| IS5 | Can the system identify packets using the EtherType Stream identification? | 5.4:d, 6.8 |  | IS: O | Yes [ ]No [ ] |

**A.2.3 Talker end system**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Item** | **Feature** | **Subclause** | **Value/Comment** | **Status** | **Support** |
| TE15 | Can the system identify packets using the EtherType Stream identification? | 5.7:d, 6.8 |  | TE: O | Yes [ ]No [ ] |

**A.2.4 Listener end system**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Item** | **Feature** | **Subclause** | **Value/Comment** | **Status** | **Support** |
| LE11 | Can the system decode packets using the EtherType Stream identification? | 5.10:d, 6.8 |  | LE: O | Yes [ ]No [ ] |

**A.2.5 Relay system**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Item** | **Feature** | **Subclause** | **Value/Comment** | **Status** | **Support** |
| RS12 | Can the system identify packets using the EtherType Stream identification? | 5.12:c, 6.8 |  | RS: O | Yes [ ]No [ ]\_\_\_\_\_\_\_\_\_\_\_1 |

**Annex D**

**D.2 Example 2: Various stack positions**

***[page 88, line 16]***

Figure D-5 illustrates relay system B in Figure D-4. As the packets enter from the left, from End System A, they pass first through a Stream Identification function (IP octuple Stream identification [6.7], or EtherType Stream identification [6.8]), which identifies the Stream. The Stream Transfer Function delivers the packet with all TSN parameters, including the stream\_handle subparameter, to the Sequence generation function (7.4.1, marked “Seq.” in Figure D-4), which adds a sequence\_number subparameter with a steadily-increasing integer sequence value (modulo the size of the packet field carrying the sequence\_number). The sequence\_number subparameter is encapsulated into the packet by the Sequence encode/decode function (7.6). A Stream Identification function (this time, Active Destination MAC and VLAN Stream identification [6.6]) modifies the two packets’ destination MAC addresses and VLANs for identification through the bridged network. Relay system B’s forwarding function then outputs the two packets on two different ports. The external form of the packets are labeled differently, as indicated by the italic numbers *26* and *31* in Figure D-4.



**Figure D-5—Protocol stack for relay system B, proxying for End System A, in Figure D-4**

**D.5 Example 5: Protocol interworking**

Figure D-10 illustrates a simple protocol interworking function in one port of a relay system. In this example, two different encapsulation schemes **1** and **2** are used for the two legs of the Stream Transfer Function, so that packets are transformed from using one encapsulation to using the other encapsulation as they pass through the port. No additional functions, e.g. a Sequence recovery function (7.4.2) are shown,although they would be perfectly admissible. If this were a port of a bridge attached to an end system, encapsulation **1** could be the Active Destination MAC and VLAN Stream identification (6.6), and encapsulation **2** could be the IP octuple Stream identification (6.7) or the EtherType Stream identification function (6.8). The net result for the end system could be to convert a specific unicast Internet Protocol Stream to use a specific multicast destination address and VLAN, in order to direct the packet through a specific path through the bridged network. Presumably, a similar interworking pair at the other end of the Stream would restore the packet to its original destination MAC address and VLAN.