Technical Specification

MEF 49

Service Activation Testing Control Protocol and PDU Formats

October 2014

Table of Contents

1. List of Contributing Members 1
2. Abstract 1
3. Terminology and Acronyms 2
4. Scope 9
5. Compliance Levels 9
6. Numerical Prefix Conventions 10
7. Introduction 11
7.1 SAT Test PDUs 11
7.2 Generator Test Function and Collector Test Function 11
7.3 Use of FL-PDU and FD-PDU 17
7.3.1 SAT Test PDU Mix 18
7.4 SAT Control Protocol 18
7.4.1 Initiate Session Request Message 22
7.4.2 Initiate Session Response Message 22
7.4.3 Start Session Request Message 22
7.4.4 Start Session Response Message 23
7.4.5 Stop Session Request Message 23
7.4.6 Stop Session Response Message 23
7.4.7 Abort Session Request Message 23
7.4.8 Abort Session Response Message 23
7.4.9 Get Session Status Request Message 24
7.4.10 Get Session Status Response Message 24
7.4.11 Fetch Session Results Request Message 24
7.4.12 Fetch Session Result Response Message 24
7.4.13 Delete Session Request Message 24
7.4.14 Delete Session Response Message 25
7.5 SAT Protocol Flow 25
7.6 GTF and CTF Processing Requirements 27
8. SAT Test PDU Formats 29
8.1 Frame Loss PDU 29
8.1.1 Reserved 1 31
8.1.2 Version 31
8.1.3 OpCode 31
8.1.4 Flags 31
8.1.5 TLV Offset 31
8.1.6 Reserved 2 32
8.1.7 Optional TLVs 32
8.1.7.1 TLV Type 32
8.1.7.2 TLV Length 33
8.1.7.3 TLV Value 33
8.1.7.4 Data TLV 33
8.2 FD-PDU 34
8.2.1 Interleaving FL-PDU and FD-PDU 34
9. SAT Control Protocol State Machines 36
9.1 State Machine Descriptions 36
9.2 Controller End State Machines 39

9.2.1 Controller End Global State Machine 41
9.2.1.1 Controller End Global State Machine Events 41
9.2.1.2 Controller End Global State Machine Actions 41
9.2.1.3 Controller End Global State Machine Functions 42
9.2.1.4 Controller End Global State Machine 42
9.2.2 Controller End Session State Machine 44
9.2.2.1 Controller End Session State Machine Events 44
9.2.2.2 Controller End Session State Machine Actions 46
9.2.2.3 Controller End Session State Machine Functions 48
9.2.2.4 Controller End Session State Machine 49
9.3 Responder End State Machines 59
9.3.1 Responder End Global State Machine 60
9.3.1.1 Responder End Global State Machine Events 60
9.3.1.2 Responder End Global State Machine Actions 61
9.3.1.3 Responder End Global State Machine Functions 62
9.3.1.4 Responder End Global State Machine 62
9.3.2 Responder End Session State Machines 64
9.3.2.1 Responder End Session State Machine Events 64
9.3.2.2 Responder End Session State Machine Actions 65
9.3.2.3 Responder End Session State Machine 67
10. SAT Control Protocol PDU 72
10.1 SAT Frame Set 72
10.2 SAT Control Protocol Message 74
10.2.1 MEL 75
10.2.2 Version 75
10.2.3 OpCode 75
10.2.4 Flags 76
10.2.5 TLV Offset 76
10.2.6 Message Type 76
10.2.7 Test Session ID 77
10.2.8 Optional TLVs 77
10.2.8.1 TLV Type 78
10.2.8.2 TLV Length 78
10.2.8.3 TLV Value 79
10.3 SAT Control Protocol Response 79
10.3.1 MEL 79
10.3.2 Version 79
10.3.3 OpCode 80
10.3.4 Flags 80
10.3.5 TLV Offset 80
10.3.6 Message Type 80
10.3.7 Test Session ID 81
10.3.8 Response Code 81
10.3.9 Optional TLVs 83
10.3.9.1 TLV Type 84
10.3.9.2 TLV Length 85
10.3.9.3 TLV Value 85
10.4 SAT TLVs 85
10.4.1 Tests and TLV Type Use 89
10.4.2 SAT TLV Subtypes 91

10.4.2.1 Measurement Type ... *91*
10.4.2.2 MAC Address and Destination MAC Address ... *91*
10.4.2.3 Green PCP ... *94*
10.4.2.4 Yellow PCP ... *94*
10.4.2.5 Duration ... *95*
10.4.2.6 EtherType and SAP ... *95*
10.4.2.7 Frame Length ... *96*
10.4.2.8 Frame Pattern ... *97*
10.4.2.9 Frame Quantity ... *98*
10.4.2.10 Frame Interval ... *99*
10.4.2.11 Green Rate ... *99*
10.4.2.12 Yellow Rate ... *99*
10.4.2.13 Yellow Frame Quantity ... *100*
10.4.2.14 Measured Rate Duration ... *100*
10.4.2.15 Test Session Status ... *101*
10.4.2.16 Yellow DEI ... *101*
10.4.2.17 Rate Type ... *102*
10.4.2.18 Measured Rate Green Bits ... *103*
10.4.2.19 Measured Rate Yellow Bits ... *103*
11. References ... **104**
12. Appendix I Interaction Diagrams ... **105**
12.1 Key ... 105
12.2 Test Configurations ... 105
12.2.1 Forward Test Session ... 106
12.2.1.1 Session Initiation ... *106*
12.2.1.2 Running a Test ... *107*
12.2.1.3 Abort Session ... *108*
12.2.2 Backward Test Session ... 109
12.2.2.1 Session Initiation ... *109*
12.2.2.2 Running a Session ... *110*
12.2.2.3 Abort Session ... *112*

List of Figures

Figure 1 Generator Test Function Relationship to MEPs in an ETE-A at a UNI-N 12
Figure 2 Generator Test Function Relationship to MEPs in an ETE-I 13
Figure 3 Collector Test Function Relationship to MEPs in an ETE-A.. 14
Figure 4 Collector Test Function Relationship to MEPs in an ETE-I 15
Figure 5 Generator Test Function in an ETE-A at an ENNI 16
Figure 6 Collector Test Function in an ETE-A at an ENNI 17
Figure 7 FL-PDU Relationship to FD-PDU 18
Figure 8 SAT Control Protocol Scope 19
Figure 9 SAT PDU Step Diagram 20
Figure 10 SAT Control Protocol PDU Protocol Exchange Example 25
Figure 11 FL-PDU Format 29
Figure 12 FL-PDU in an Ethernet Frame 30
Figure 13 FL-PDU TLV Format 32
Figure 14 State Machine Figure Conventions 38
Figure 15 Controller End State Machines Overview 40
Figure 16 Controller End Global State Machine 43
Figure 17 Controller End Session State Machine – Create Stage 50
Figure 18 Controller End Session State Machine – Pending Stage 51
Figure 19 Controller End Session State Machine – Start Stage 52
Figure 20 Controller End Session State Machine – Run Stage 53
Figure 21 Controller End Session State Machine – Stop Stage 54
Figure 22 Controller End Session State Machine – Fetch Stage 55
Figure 23 Controller End Session State Machine – Delete Stage 56
Figure 24 Controller End Session State Machine – Abort Stage 57
Figure 25 Controller End Session State Machine – Status stage 58
Figure 26 Responder End State Machines Overview 60
Figure 27 Responder End Global State Machine 63
Figure 28 Responder End Session State Machine – Start stage 68
Figure 29 Responder End Session State Machine – Run stage 69
Figure 30 Responder End Session State Machine – Fetch stage 70
Figure 31 Responder End Session State Machine – Delete stage 71
Figure 32 SAT FS VLAN ID Translation 74
Figure 33 Generic SAT Control Protocol Message Format 75
Figure 34 Optional TLV Format 77
Figure 35 Generic SAT Control Protocol Response Format 79
Figure 36 Optional TLV Format 84
Figure 37 SAT TLV Format 85
Figure 38 Interactive Drawings Key 105
Figure 39 Initiating a Session for a Forward Test 107
Figure 40 Running a Test Session for a Forward Test 108
Figure 41 Aborting a Session for a Forward Test 109
Figure 42 Initiating a Session for a Backward Test 110
Figure 43 Running a Session for a Backward Test 111
Figure 44 Aborting a Test Session for a Backward Test 113

List of Tables

Table 1 Terminology and Acronyms 9
Table 2 Numerical Prefix Conventions 10
Table 3 SAT FL-PDU Types 33
Table 4 SAT Control Protocol SCM Message Types 77
Table 5 SAT Control Protocol SCM TLV Types 78
Table 6 SAT Control Protocol SCR Message Types 81
Table 7 MEP Control Message Response Codes 82
Table 8 SCR Optional TLV Types 84
Table 9 TLV SubTypes 88
Table 10 Initiate Session Request Message TLVs 90
Table 11 Fetch Session Results Response Message TLVs 91
Table 12 Session Status Values 101

1. List of Contributing Members

The following members of the MEF participated in the development of this document and have requested to be included in this list.

ADTRAN	Cisco Systems
ADVA Optical	Comcast
Aim Valley B.V.	JDSU
Allstream	EXFO Inc.
Broadcom	Pulse Communications
Cable Labs	PLDT Corporation
Ceragon Networks	RAD Data Communications
Ciena Corporation	Siama Systems

2. Abstract

Service Activation Testing (SAT) is defined by MEF in the SAT Technical Specification [12]. The SAT Technical Specification [12] defines testing of Network Operator services that span more than one Network Operator. The SAT Technical Specification [12] is using the processes contained in ITU-T Recommendation Y.1564 [11] as a base. Neither the SAT Technical Specification [12] nor Y.1564 [11] specify a test Protocol Data Unit (PDU). It is desirable that network operators can perform test procedures with interoperable devices from diverse test equipment vendors or NE vendors. This document standardizes the following:

- The test PDUs that are exchanged between devices during the test
- The control protocol used to configure SAT tests
- The control PDUs that are exchanged between the devices in two operators' networks.

The control protocol provides the ability to configure and control the SAT steps and to fetch test results at the completion of the test. The requirements defined within this document are based on the SAT process as defined in section 10 of the Carrier Ethernet Service Activation Testing Technical Specification [12].

3. Terminology and Acronyms

Term	Definition	Source
Access Provider	An Operator MEN that offers the Ethernet Access Service type.	MEF 33 [16]
AP	Access Provider	MEF 33 [16]
Backward	The direction of a test session from the Responder End towards the Controller End.	This document
CEN	Carrier Ethernet Network	MEF 12.2[6]
CTF	Collector Test Function	SAT Technical Specification [12]
Collector Test Function	A logical function for collecting Ethernet test measurements.	SAT Technical Specification [12]
Controller End	The endpoint that manages control procedures for tests such as SAT or Latching Loopback. It is responsible for initiating test session(s) and reporting results of the test session(s).	MEF 46 [14]
CoS	Class of Service or Classes of Service	MEF 23.1[8]
CoS Performance Objective	Performance objective specified for a CoS Label or CoS Name.	MEF 23.1[8]
DMM	Delay Measurement Message	ITU-T G.8013/Y.1731[3]
DMR	Delay Measurement Reply	ITU-T G.8013/Y.1731[3]
ENNI	External Network Network Interface	MEF 4[4]
EPCF	ETH Provider Conditioning Function	MEF 12.2[6]
ESCF	ETH Subscriber Conditioning Function	MEF 12.2[6]
ETE	Ethernet Test Equipment	MEF 46 [14]
ETE-A	Ethernet Test Equipment - Application	MEF 46 [14]
ETE-I	Ethernet Test Equipment - Instrument	MEF 46 [14]
ETE-TH	Ethernet Test Equipment-Test Head	MEF 46 [14]
ETH	Ethernet Layer or Ethernet Services Layer	MEF 4[4]
ETH-DM	Ethernet Frame Delay Measurement Function	ITU-T G.8013/Y.1731[3]

Term	Definition	Source
ETH Provider Conditioning Function	The ETH Provider Conditioning Function (EPCF) is the processing entity of the ETH Layer responsible for classification, filtering, metering, marking, policing, scheduling, shaping and, in general, conditioning flow(s) between two CENs.	MEF 12.2[6]
ETH Subscriber Conditioning Function	The ETH Subscriber Conditioning Function (ESCF) is the processing entity of the ETH Layer responsible for classification, filtering, metering, marking, policing, scheduling, shaping and, in general, conditioning the subscriber flow into and out of a Service Network Operator EFD at a UNI-N.	MEF 12.2[6]
Ethernet Frame Length	The length of a frame including the header through the FCS and excluding the interframe gap or preamble bits.	MEF 33 [16]
Ethernet Services Layer	The Ethernet Services Layer, also referred to as the ETH Layer, is responsible for the instantiation of Ethernet MAC oriented connectivity services and the delivery of Ethernet service frames presented across well-defined internal and external interfaces and associated reference points. The ETH layer is also responsible for all service-aware aspects associated with Ethernet MAC flows including operations, administration, maintenance and provisioning capabilities required to support such Ethernet connectivity services.	MEF 4[4]
Ethernet Test Equipment	A general term to include an Ethernet Test Equipment-Application, Ethernet Test Equipment-Test Head, and/or Ethernet Test Equipment-Instrument.	MEF 46 [14]

Term	Definition	Source
Ethernet Test Equipment-Application	Functionality resident in a Network Element which may include an Generator Test Function, a Collector Test Function and/or Latching Loopback function that enables the Network Element to perform Service Activation Testing and activate/deactivate loopback devices.	MEF 46 [14]
Ethernet Test Equipment-Instrument	A portable, external Ethernet testing equipment not permanently installed in the network, which may include a Generator Test Function, a Collector Test Function, and/or Latching Loopback Function that enables the ETE to perform Service Activation Testing and activate/deactivate loopback devices.	MEF 46 [14]
Ethernet Test Equipment– Test Head	An external Ethernet testing equipment permanently installed in the network, which may include a Generator Test Function, a Collector Test Function, and/or Latching Loopback Function that enables the ETE to perform Service Activation Testing and activate/deactivate loopback devices. It is not involved in the Forwarding path of services.	MEF 46 [14]
Ethernet Test Support System	A function that coordinates test activity at Ethernet Test Equipment or Network Elements.	MEF 46 [14]
Ethernet Virtual Connection	An association of two or more UNIs that limits the exchange of Service Frames to UNIs in the Ethernet Virtual Connection.	MEF 10.3[5]
ETSS	Ethernet Test Support System	MEF 46 [14]
EVC	Ethernet Virtual Connection	MEF 10.3[5]
External Network Network Interface	A reference point representing the boundary between two Network Operator CENs that are operated as separate administrative domains.	MEF 4[4]
FD-PDU	Frame Delay Protocol Data Unit	This document
FL-PDU	Frame Loss Protocol Data Unit	This document
Forward	The direction of a test session from the Controller End towards the Responder End.	This document

Term	Definition	Source
Frame Delay Protocol Data Unit	A Protocol Data Unit (PDU) used to measure Frame Delay	This document
Frame Loss Protocol Data Unit	A Protocol Data Unit (PDU) used to measure Frame Loss	This document
Generator Test Function	A logical function for generating and transmitting Ethernet Frames which can include test frames.	SAT Technical Specification [12]
GTF	Generator Test Function	SAT Technical Specification [12]
Information Rate	The average bit rate of Frames at the measurement point starting with the first MAC address bit and ending with the last FCS bit. Note: Frames can be Service Frames [5] or ENNI Frames [9].	ITU-T Y.1564 [11]
IR	Information Rate	ITU-T Y.1564 [11]
Latching Loopback	A configured function within an Ethernet Equipment where frames are returned to the entity which sent them.	MEF 46 [14]
MEF Service Activation Testing Process	A business process for testing MEF services to ensure that the service is configured according to the specification and will work to agreed performance levels (e.g., SLAs). This process occurs before the service is deployed to the customer.	SAT Technical Specification [12]
MEG	Maintenance Entity Group	ITU-T G.8013/Y.1731[3]
MEL	MEG Level	ITU-T G.8013/Y.1731[3]
MEP	MEG End Point	ITU-T G.8013/Y.1731[3]
One-way	A measurement performed in the Forward or Backward direction. For example from MEP A to MEP B or from MEP B to MEP A.	MEF 35[10]
Operator Virtual Connection	Operator Virtual Connection, an association of OVC End Points.	MEF 26.1[9]
OVC	Operator Virtual Connection	MEF 26.1[9]
PDU	Protocol Data Unit	NA
PRBS	Pseudo Random Bit Sequence	NA

Term	Definition	Source
Responder End	The endpoint that is not managing the test session. It is responsible for responding to messages from the Controller End.	MEF 46 [14]
SAT	Service Activation Test	SAT Technical Specification [12]
SAT Control Protocol	Service Activation Test Control Protocol	This document
SAT Frame Set	A set of frames generated by a GTF or counted by a CTF within a particular EI, identified by containing the same CE-VLAN ID and/or S-VLAN ID, or by being untagged. Received SAT Control Frames may be consumed by a MEP, before they reach the CTF; however, they are considered to belong to a SAT FS according to the tags they would have when passing through the CTF, if the consuming MEP did not exist. Similarly, SAT Control Frames sent by a MEP are considered to belong to a SAT FS according to the tags they would need to have been generated with at the GTF, such that they would be transmitted with the same tags if the MEP did not exist.	This Document
SAT FS	SAT Frame Set	This document
SCM	Service Activation Test Control Protocol Message	This document
SCR	Service Activation Test Control Protocol Response	This document
Service Acceptance Criteria	The pass/fail limits for the various performance parameters that are measured during the Service Activation Test.	SAT Technical Specification [12]

Term	Definition	Source
Service Activation Measurement Point	A Service Activation Measurement Point is a reference point in the network, placed at the ingress of an ETH Subscriber/Provider Conditioning Function in the case of a Generator Test Function or at the egress of an ETH Subscriber/Provider Conditioning Function in the case of a Collector Test Function, at which performance reference events can be observed and measured during the Service Activation Testing process.	SAT Technical Specification [12]
Service Activation Testing	The process of executing a collection of test procedures to be applied to a given traffic entity (e.g., EVC, OVC, etc.) in order to collect behavioral information about the traffic and compare this with predefined expectations.	SAT Technical Specification [12]
Service Activation Test Control Protocol	The messages and responses used by the SAT Controller End and the SAT Responder End to coordinate a SAT measurement.	This document
Service Activation Test Control Protocol Message	Used by the SAT Controller End to send requests to the SAT Responder End.	This document
Service Activation Test Control Protocol Response	Used by the SAT Responder End to send responses to the SAT Controller End.	This document
Service Activation Test Methodology	A methodology for performing Ethernet Service turn-up testing using defined benchmark tests while measuring various performance parameters.	SAT Technical Specification [12]
Service Activation Test Record	A report of test results for a new Ethernet Service. The results show if the service met the applicable performance objectives or Service Acceptance Criteria.	SAT Technical Specification [12]
Service Configuration and Activation	A high level business process, defined by the TMF, which encompasses operational processes for the allocation, implementation, configuration, activation and testing of specific services to meet customer requirements.	TM Forum

Term	Definition	Source
Service Level Agreement	The contract between the Subscriber or Operator and Service Network Operator specifying the agreed to service level commitments and related business agreements.	MEF 10.3[5]
SLA	Service Level Agreement	MEF 10.3[5]
Service Measurement Point	A Service Measurement Point is a well-defined point in the network at which performance reference events can be observed and measured.	SAT Technical Specification [12]
Service Provider	The organization providing UNI to UNI Ethernet Service(s).	MEF 33 [16]
SP	Service Provider	MEF 33 [16]
SOAM	Service Operations, Administration and Maintenance	MEF 17[7]
Test Session	A test defined by a single set of parameters between a Controller End and a Responder End started by an Initiate Session Message, and with the results collected using the Fetch Session Message.	This document
TLV	Type, Length, Value	G.8013/Y.1731[3]
Traffic Conditioning Point	Corresponds to an ESCF (Ethernet Subscriber Conditioning Function).	MEF 17[7]
Two-way	A measurement of the performance of frames that flow from the Controller MEP to Responder MEP and back again.	MEF 35[10]
ULR	Utilized Line Rate	ITU-T Y.1564 [11]
UNI	User Network Interface	MEF 10.3[5]
User Network Interface (UNI)	The physical demarcation point between the responsibility of the Service Network Operator and the responsibility of the Subscriber.	MEF 10.3[5]
UNI-C	User Network Interface-Client	MEF 4[4]
UNI-Client	A compound architectural component of a CEN that represents all of the functions required to connect a subscriber to a CEN.	MEF 4[4]
UNI-N	User Network Interface-Network	MEF 4[4]
UNI-Network	A compound architectural component of a CEN that represents all of the functions required to connect a CEN to a CEN subscriber.	MEF 4[4]

Term	Definition	Source
Utilized Line Rate	The average bit rate of the Ethernet line at the measurement point, including the bits a) allocable to the minimum-duration period of each Inter-Packet gap (but not the number of bits allocable to the part of each Inter-Packet gap longer than the minimum), b) in the preamble, c) in the start of frame delimiter and d) in the Ethernet Service Frame starting with the first MAC address bit and ending with the last FCS bit.	ITU-T Y.1564 [11]

Table 1 Terminology and Acronyms

4. Scope

This document defines the SAT Test PDU and SAT Control Protocol PDU formats and the control protocol used by Ethernet Test Equipment (ETE) when testing between two Network Operators across an ENNI. SAT Test PDU formats are defined to support frame loss and frame delay measurements. Support for two SAT Test PDUs is required by this document. The SAT Control Protocol PDUs are carried within SOAM frames between two MEPs used to support SAT. The control protocol communicates SAT attributes between the Controller End and the Responder End of the SAT, identifies what SAT steps to perform and communicates test results between the two end points at the completion of the test.

5. Compliance Levels

The requirements that apply this document are specified in the following Sections. Items that are REQUIRED (contain the words MUST or MUST NOT) will be labeled as [Rx]. Items that are RECOMMENDED (contain the words SHOULD or SHOULD NOT) will be labeled as [Dx]. Items that are OPTIONAL (contain the words MAY or OPTIONAL) will be labeled as [Ox].

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in RFC 2119 [2]. All key words use upper case, bold text to distinguish them from other uses of the words. Any use of these key words (e.g., may and optional) without [Rx], [Dx] or [Ox] is not normative.

6. Numerical Prefix Conventions

This document uses the prefix notation to indicate multiplier values as shown in Table 2.

Decimal		**Binary**	
Symbol	**Value**	**Symbol**	**Value**
k	10^3	Ki	2^{10}
M	10^6	Mi	2^{20}
G	10^9	Gi	2^{30}
T	10^{12}	Ti	2^{40}
P	10^{15}	Pi	2^{50}
E	10^{18}	Ei	2^{60}
Z	10^{21}	Zi	2^{70}
Y	10^{24}	Yi	2^{80}

Table 2 Numerical Prefix Conventions

7. Introduction

The SAT Technical Specification [12] includes the recommended methodologies for SAT. It does not define the format of the PDU used to perform SAT. ETE vendors have implemented their own version of a PDU for SAT and typically the PDUs and procedures from two ETE vendors are not interoperable. To resolve this issue, a set of standard SAT Test PDUs is defined in section 7.1. These PDUs are used to perform SAT between Network Operators. This allows ETE vendors to be compatible with each other.

In addition to the SAT Test PDUs, a method of configuring the test and exchanging test results is needed. The SAT Control Protocol provides this method. With the use of the SAT Test PDUs and SAT Control Protocol, multiple Network Operators are able to perform SAT without the need for loopbacks and for technicians or systems to exchange test results after a test is complete. The SAT Control Protocol is defined in section 7.4.

7.1 SAT Test PDUs

To support the processes defined in the SAT document, two SAT Test PDUs are used. One SAT Test PDU, known as the Frame Loss PDU (FL-PDU), is used to perform frame loss and "throughput" measurements. This is a new PDU defined within this document in section 8.1. The other SAT Test PDU, known as the Frame Delay PDU (FD-PDU), is used to perform frame delay measurements. The FD-PDUs are the DMM PDU and DMR PDU as defined in G.8013/Y.1731[3]. The use of the FL-PDUs and the FD-PDUs are explained in section 7.3. By using these SAT Test PDUs, ETE vendors are able to perform tests with each other and SAT can be performed between multiple Network Operators without the need for loopbacks and additional testing.

Note: The FL-PDU is not used for L2CP testing. If L2CP testing is required, a standard L2CP frame is used.

7.2 Generator Test Function and Collector Test Function

The SAT Technical Specification [12] defines a Generator Test Function (GTF) and a Collector Test Function (CTF). A GTF transmits test frames and a CTF receives test frames. This concept is used within this document as well. In addition, this document uses MEPs to perform frame delay measurements. Figure 1 provides an architectural view of the location of the GTF and corresponding MEPs.

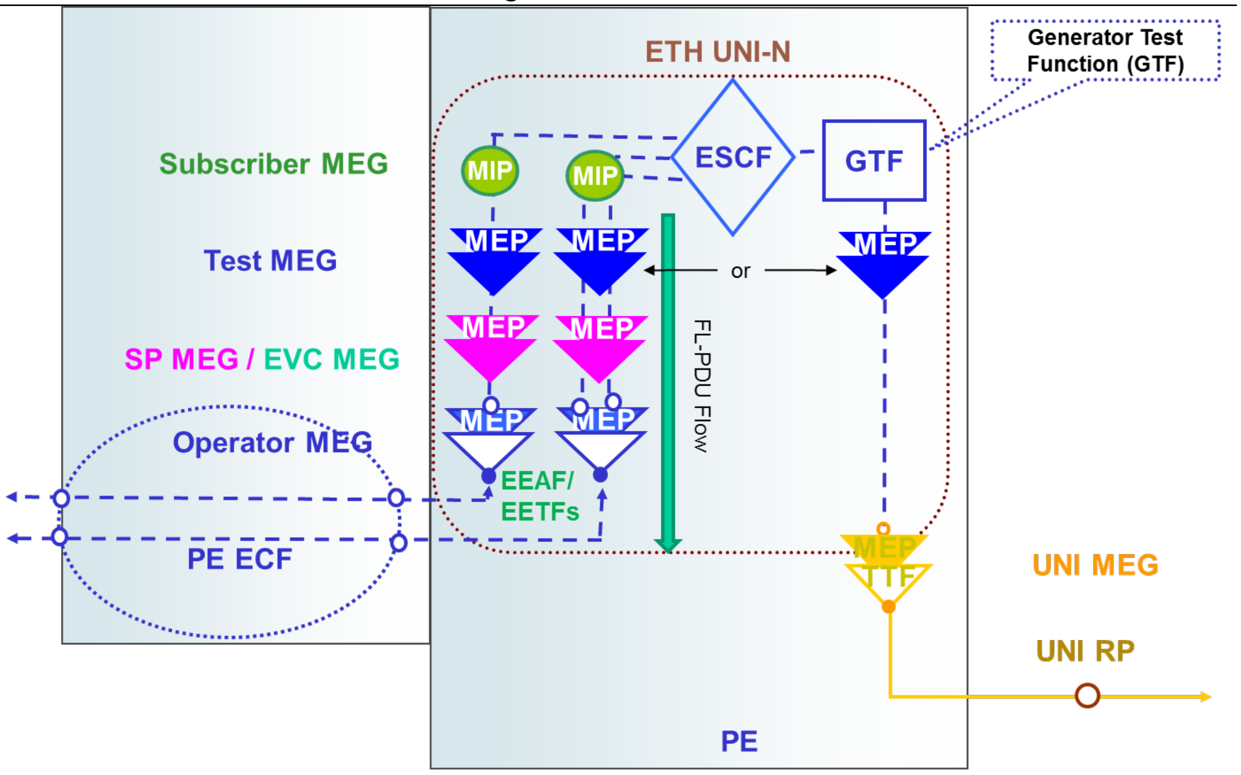


Figure 1 Generator Test Function Relationship to MEPs in an ETE-A at a UNI-N

Note: The "or" directly under the ESCF in Figure 1 is there to indicate that the MEPs shown on the left hand side of the ESCF or the MEP on the right hand side of the ESCF can be used to support the Controller End or Responder End Frame Delay measurements. The MEPs to the left of the ESCF are used to perform measurements to the far-end. The MEP on the right hand side of the ESCF is used to perform measurements to the UNI-C. This convention applies to similar figures within this document.

A GTF in an ETE-A should be similar to an instrument connected to the UNI-N and hence it is located on the UNI-C side of the Ethernet Subscriber Conditioning Function (ESCF).

This relationship is slightly different in the case of an ETE-I or ETE-TH.

Figure 2 shows an example of an ETE-I implementation where the GTF and the MEP are contained in the ETE-I.

An ETE-TH implementation looks similar to the ETE-I implementation and an example is not provided within this document.

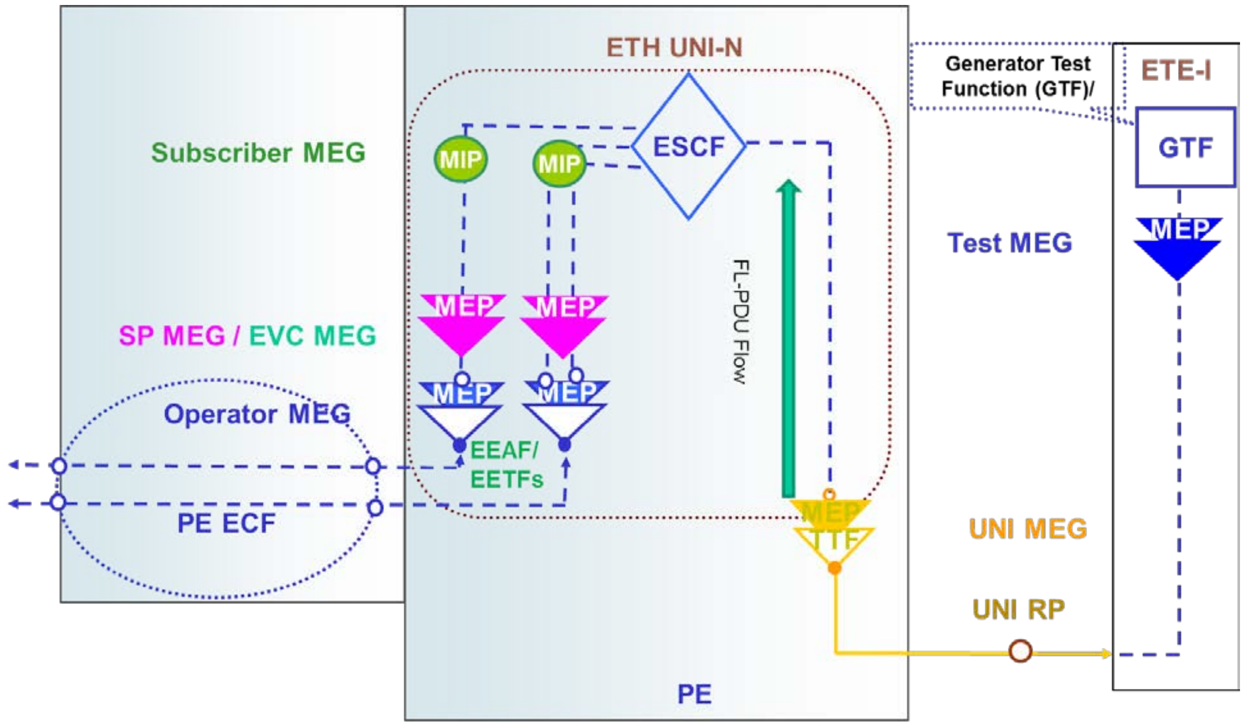


Figure 2 Generator Test Function Relationship to MEPs in an ETE-I

Figure 3 and Figure 4 show the relationship of the CTF to the MEPs. In Figures 1-4, MEPs at the Test MEG, SP MEG, EVC MEG or Operator MEG can be used to support the control protocol, instantiate the GTF and CTF and/or perform the delay measurements.

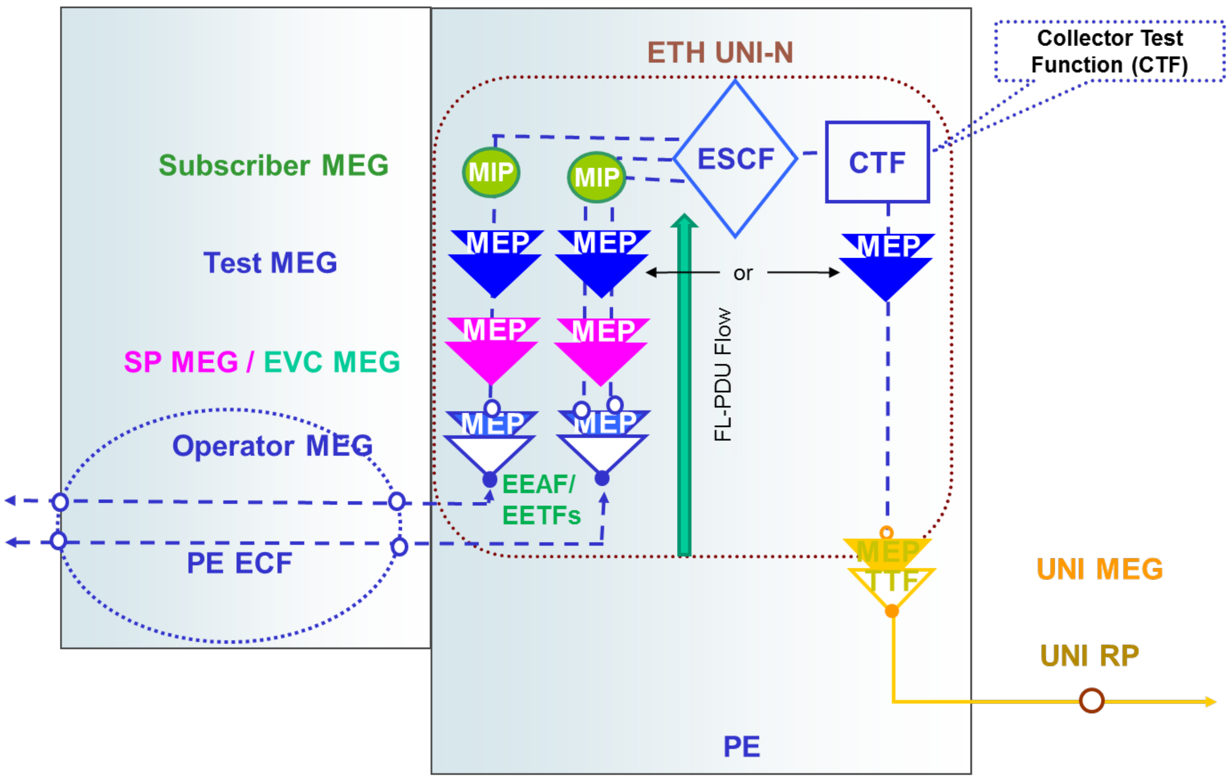


Figure 3 Collector Test Function Relationship to MEPs in an ETE-A

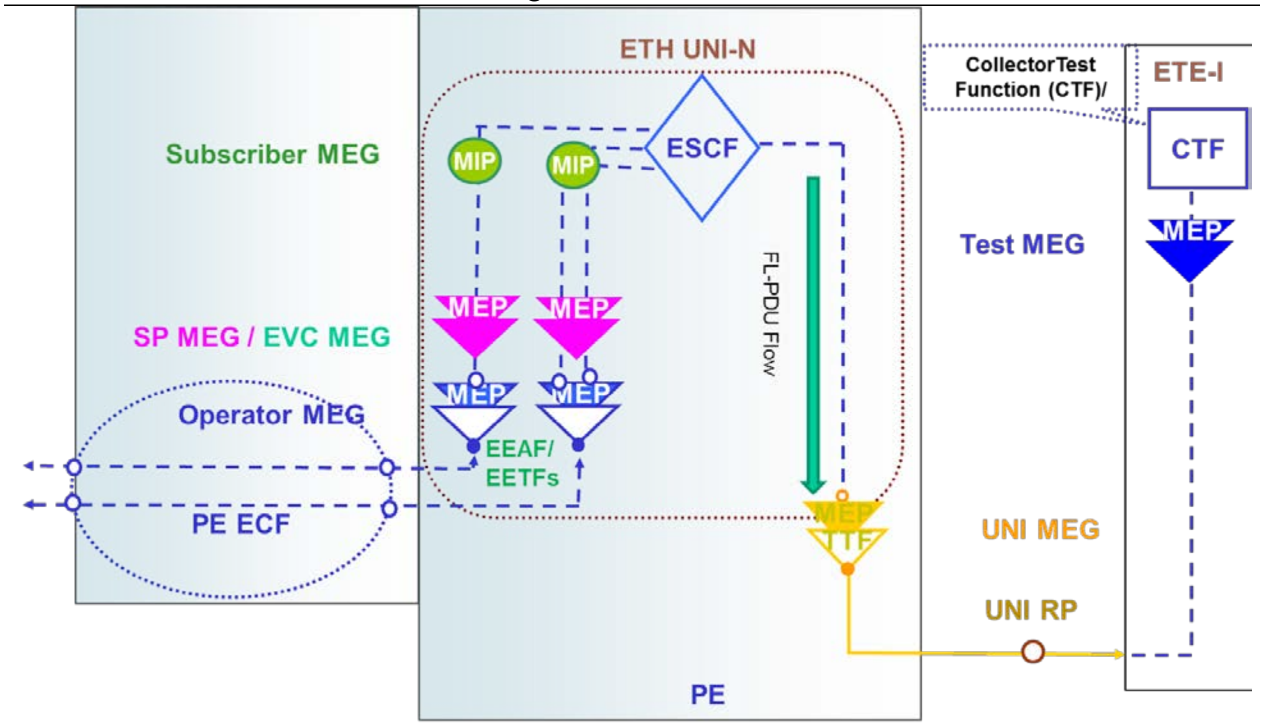


Figure 4 Collector Test Function Relationship to MEPs in an ETE-I

The relationship between the CTF and GTF, the MEPs and the EPCF are similar at an ENNI. An example is shown in Figure 5.

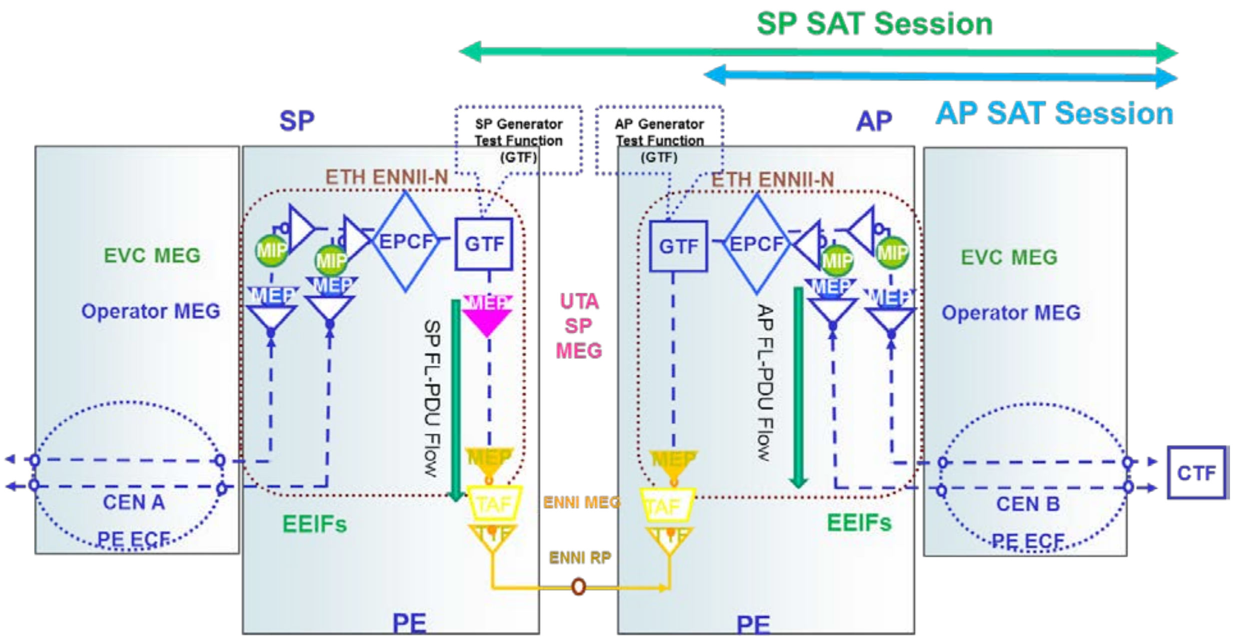


Figure 5 Generator Test Function in an ETE-A at an ENNI

As seen in Figure 5, the Service Provider (SP) or Access Provider (AP) may have a GTF function at the ENNI. In the case of the SP, the GTF is located so that frames generated by it do not pass through the EPCF on the SP side of the ENNI and do pass through the EPCF on the AP side of the ENNI. The AP's GTF is located so that frames generated by it do pass through the EPCF on the AP side of the ENNI.

Figure 6 shows the locations of the respective CTFs at the ENNI. Similar to the GTF, the SP's CTF is located so that frames do not pass through the EPCF on the SP side of the ENNI but have passed through the EPCF on the AP side of the ENNI. The AP's CTF is located so that frames do pass through the EPCF on the AP's side of the ENNI. In Figures 5 and 6, MEPs at the SP

MEG, EVC MEG or Operator MEG can be used to support the control protocol, instantiate the GTF and CTF and/or perform the delay measurements.

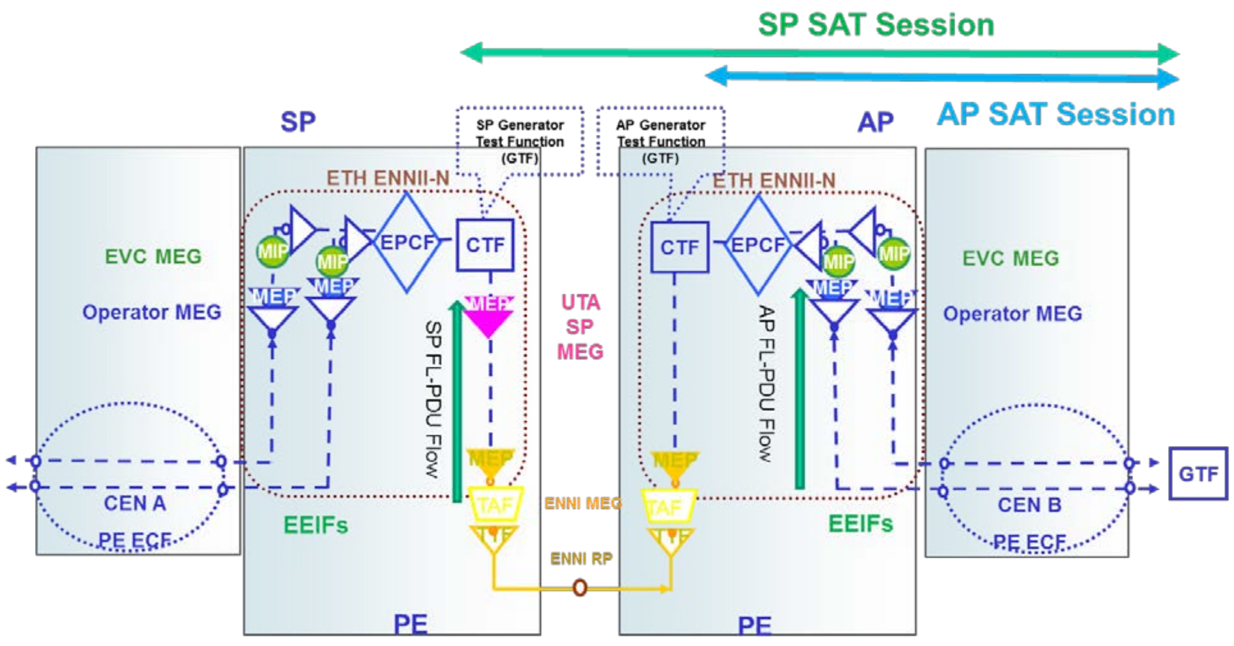


Figure 6 Collector Test Function in an ETE-A at an ENNI

When two uni-directional measurements are being performed simultaneously in opposite directions, there is a GTF and CTF at each end of the OVC. This relationship is detailed in the SAT Technical Specification [12].

7.3 Use of FL-PDU and FD-PDU

The two SAT Test PDUs defined in this document, the FL-PDU and the FD-PDU, are used to perform SAT as defined in the SAT Technical Specification [12]. SAT is performed using the FL-PDU unless delay measurements are required, then SAT is performed using both FL-PDUs and FD-PDUs. MEPs are used to measure delay using DMM and DMR frames. MEPs are located where frame delay measurements do not include the ESCF to reduce the duplication of delay measurement capabilities within devices (DMM/DMR for SOAM and separate SAT delay PDU would be required) and since the impact of the ESCF on frame delay is minimal. The FD and IFDV are not expected to be impacted by the ESCF if the IR is constant as is done with a SAT test. The MEPs that are created to support the SAT Control Protocol can also be used to perform the frame delay measurements. If these MEPs exist within the ETE then they might not be the same MEPs that are used to monitor the specific service.

7.3.1 SAT Test PDU Mix

When both FL-PDUs and FD-PDUs are required to perform the one-way measurement defined in the SAT Technical Specification [12], the GTF transmits the FL-PDUs at a rate required to perform the specific measurement. At the same time, the MEPs at the appropriate MEG transmit the DMM frames at some regular interval that is expected to be greater than the FL-PDU interval. See MEF 35 [10] section 7.1 for more detail on the use of DMM/DMR for Frame Delay measurements. The relationship between FL-PDUs and FD-PDUs is shown in Figure 7. The indicated time interval between FD-PDUs is for example only. The FD-PDUs are sent in addition to the FL-PDUs and the additional synthetic traffic introduced by the FD-PDUs in this scenario is not counted as part of the Loss or Throughput measurement test being conducted between GTF and CTF. Note that the Ethernet Test Support System (ETSS) is responsible for coordination of FL-PDUs and FD-PDUs. If the ETSS is non-automated, the FD-PDUs can be configured manually.

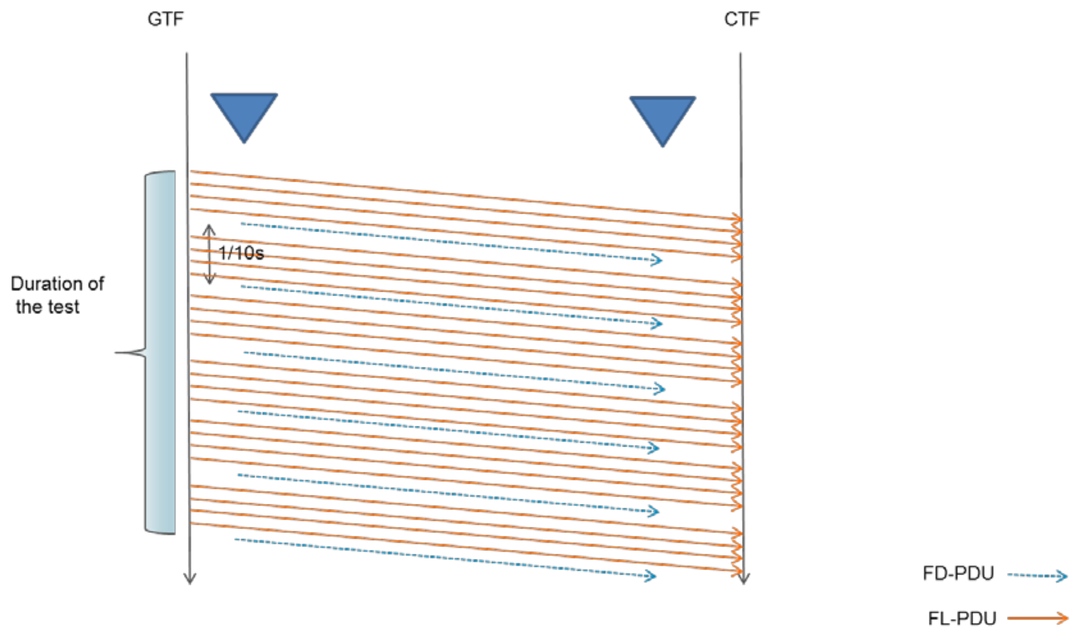


Figure 7 FL-PDU Relationship to FD-PDU

If delay measurements are not required for the test, FD-PDUs are not sent.

7.4 SAT Control Protocol

The SAT Control Protocol is used to manage test sessions between two different ETEs and is specified in this document. The SAT Control Protocol is peered between the pair of MEPs located at either ends of OVC under test. A test session is defined within the scope of this document as the exchange of control protocol messages needed to perform a uni-directional measurement with a defined set of parameters and retrieve the measurement results. One or more test sessions are used for each of the tests defined in section 10 of the SAT Technical Specification

[12]. Multiple test sessions can be used concurrently between the same pair of ETEs. A test session is uni-directional and has two ends, the Controller End and the Responder End, as shown in Figure 8.

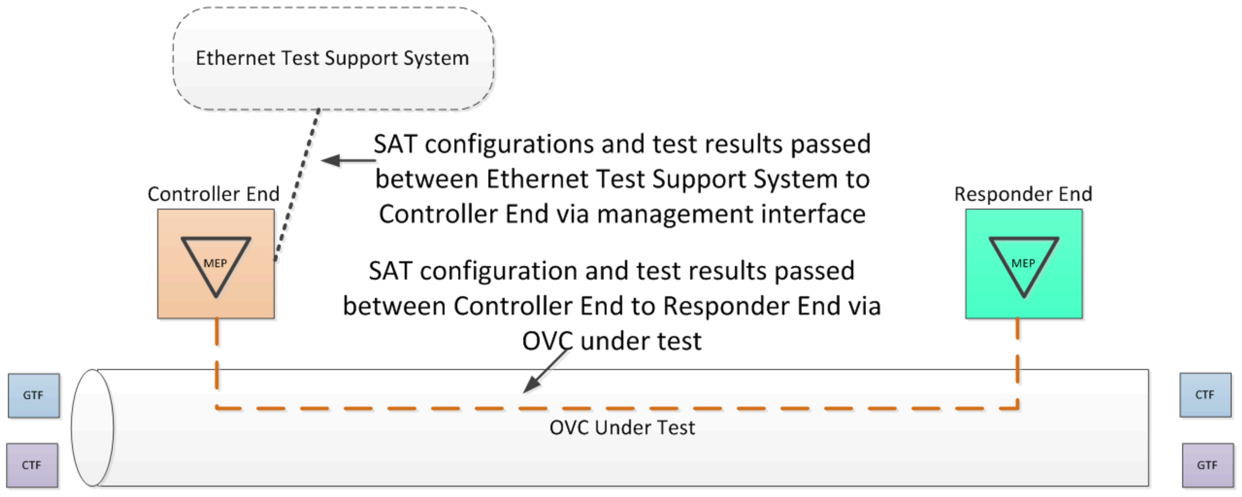

Ethernet Test Support System
SAT configurations and test results passed between Ethernet Test Support System to Controller End via management interface
Controller End
MEP
Responder End
MEP
SAT configuration and test results passed between Controller End to Responder End via OVC under test
GTF
CTF
OVC Under Test
CTF
GTF

Figure 8 SAT Control Protocol Scope

The Controller End is responsible for initiating the test session and for providing the session attributes to the Responder End. There are three methods through which the Responder End MAC address may become known to the Controller End. The first is through out-of-band means. The second is through the use of Link Trace or multicast Loopback (LBM/LBR) as defined in [3], if supported. The third is through the CCM database as defined in [3] if CCMs are being exchanged between the two MEPs.

Each Test Session created at a Controller End MEP is assigned a unique numeric ID in the range 1 to (2^{32})-1 known as the Test Session ID. The Test Session ID is unique across all test sessions for which the MEP is acting as the Controller End, except where multiple MEPs share a single MAC address. If two MEPs share a MAC address, then they must not create test sessions with the same Test Session ID because different test sessions may interfere with each other. If MAC addresses are not shared, the Test Session ID can be duplicated.

The Controller End also supports the GTF and/or CTF as applicable to the test session. The Responder End responds to requests from the Controller End, and configures its GTF or CTF as requested by the Controller End. If the Responder End does not support the attributes requested by the Controller End, it denies the request with a Response Code indicating what cannot be supported. The Controller End can then request another test session with different test attributes.

Note: The scope of the SAT Control Protocol is shown by the dashed orange line in Figure 8. The Scope of the SAT itself is the black tube representing the OVC.

When testing between two different ETEs, the SAT control protocol is used by the Controller End to:

- Configure the GTF and/or CTF at the Responder End of the OVC
- To configure the test attributes such as number of frames, frame size, duration of the test session at the Responder End
- To start and stop test sessions
- To retrieve test result data from the Responder End

The SAT control protocol uses SAT Control Protocol PDUs between two MEPs contained in ETE-As, ETE-Is, or ETE-THs. The different message types are identified by SAT Message Type within the SAT Control Protocol Message or SAT Control Protocol Response.

Note: The MEPs used to exchange SAT Control PDUs are not required to support nor prohibited from supporting functions like CCM, Loopback, Link Trace, etc., for the purpose of supporting the SAT PDU functions.

An example of the SAT process including the interaction between the Ethernet Test Support System (ETSS), the ETE acting as the Controller End and the ETE acting as the Responder End is detailed below.

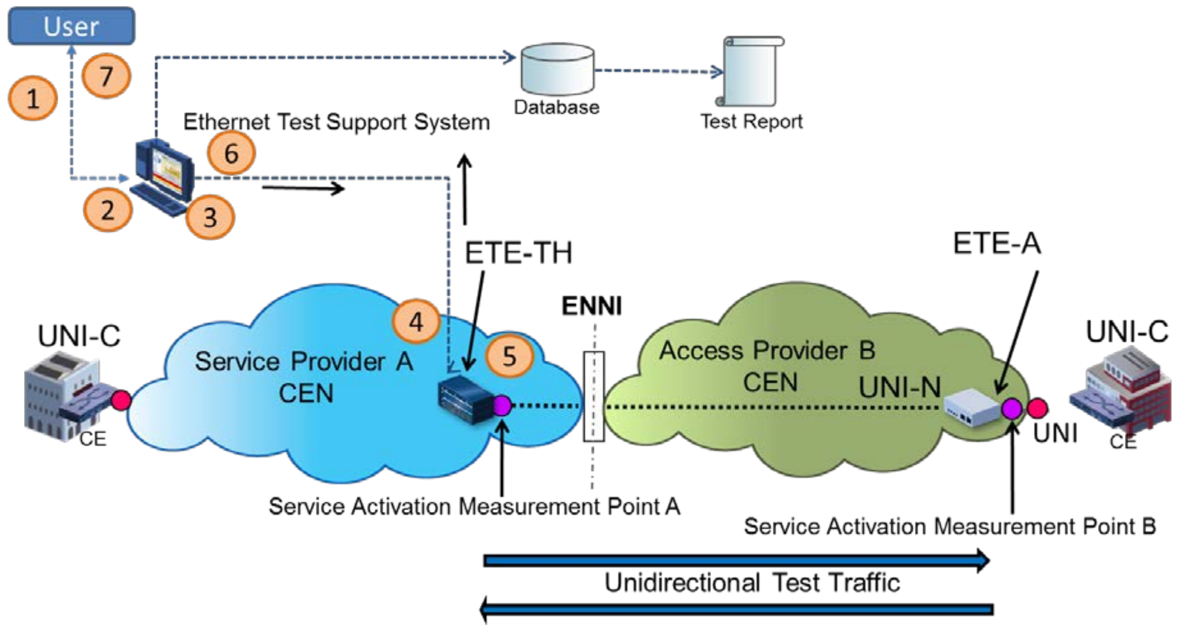

User
1
7
2
3
6
Ethernet Test Support System
Database
Test Report
ETE-TH
ETE-A
ENNI
UNI-C
UNI-C
4
5
Service Provider A CEN
Access Provider B CEN
UNI-N
UNI
CE
CE
Service Activation Measurement Point A
Service Activation Measurement Point B
Unidirectional Test Traffic

Figure 9 SAT PDU Step Diagram

1. The user executes a "Run SAT" command after specifying a minimal set of parameters (at a minimum, the target device).

2. The ETSS divides this into a set of SAT tests (as defined in [12] that may be run consecutively. For more information on the tests and their execution model, please refer to the SAT Technical Specification [12].

3. For each SAT test, the ETSS determines the test sessions that will be needed and the parameters for each one; and whether the test sessions are run concurrently or consecutively. For example, a unicast delivery test requires 2 test sessions (one in each direction) that are run concurrently, while a CIR step test requires a number of test sessions for each direction, with different transmit rates, which are run consecutively.

4. For each test session required, the ETSS requests that the SAT Controller End create the session and then start it.

   a) If the test session requires loss measurements, the Controller End uses the control protocol defined in this document to set up and start the test at the Responder End.

   b) If the test session requires frame delay measurements, these are performed using FD-PDUs. In this case the MEP at the Controller End sends DMMs and the MEP at the Responder End replies with DMRs. The SAT Control Protocol is not used to configure or perform delay measurements. The ETSS is responsible for configuration of the FD-PDUs and for retrieving results from the delay measurements.

5. Once the test has run, the Controller End retrieves the results from the Responder (using the SAT Control Protocol), combines them with its own results and returns these to the ETSS.

6. Once all of the tests have completed, the ETSS combines the results from all of the test sessions that comprise a given SAT Test, to determine whether the test has passed and to build the test report.

7. The ETSS returns the combined results to the user in a test report, as per section 11 of the SAT Technical Specification [12].

Note that in the case of an ETE-I, the ETSS is likely in the same device as the Controller End, whereas for an ETE-A or ETE-TH, it may be in a different device. This document does not specify the interface between the Controller End and the ETSS, or the manageability model between the ETSS and the user.

The SAT Control Protocol messages are defined below:

- Initiate Session Request Message
- Initiate Session Response Message
- Start Session Request Message
- Start Session Response Message
- Stop Session Request Message
- Stop Session Response Message
- Abort Session Request Message
- Abort Session Response Message
- Get Session Status Request Message
- Get Session Status Response Message
- Fetch Session Results Request Message
- Fetch Session Results Response Message
- Delete Session Request Message
- Delete Session Response Message

Request messages are carried within Service Activation Testing Control Protocol Messages (SCMs). Response messages are carried within Service Activation Testing Control Protocol Responses (SCRs). Response messages are sent in response to a Request message. Stop Session and Abort Session Response Messages can be sent without a corresponding Request Message. The use of each of these messages is summarized in the following sub-sections.

7.4.1 Initiate Session Request Message

The Initiate Session Request Message is used by the Controller End to initiate a test session. It contains all of the parameters of the test session including information such as whether the Responder End needs to instantiate a CTF or GTF, how many and what type of frames to send and how often. A full list of parameters is detailed in section 10.

7.4.2 Initiate Session Response Message

The Initiate Session Response Message is used by the Responder End to inform the Controller End whether it can comply with the Initiate Session Request Message.

7.4.3 Start Session Request Message

The Start Session Request Message is used by the Controller End to start the accepted test session at the Responder End. It is only used for Backward tests (Responder End to Controller End). In this case, the Controller End enables the CTF (i.e., starts counting frames) for this test session before sending the Start Session Request Message, and the Responder End enables the GTF (i.e., starts transmitting frames) upon receiving it.

Note: in a Forward test (Controller End to Responder End), the Responder enables the CTF when the session is initiated, so no Start Session Request message is needed.

7.4.4 Start Session Response Message

The Start Session Response Message is used by the Responder End to acknowledge the Start Session Request received from the Controller End.

7.4.5 Stop Session Request Message

The Stop Session Request Message is used by the Controller End to stop a test session. This is used at the completion of a test session.

7.4.6 Stop Session Response Message

The Stop Session Response Message is used by the Responder End for two purposes. These are:

- In a Forward test to acknowledge the Stop Session Request Message received from the Controller End.
- In a Backward test to indicate that all requested frames have been transmitted by the GTF at the Responder End. This does not have a corresponding Stop Session Request Message.

7.4.7 Abort Session Request Message

The Abort Session Request Message is used by the Controller End to abort a test session. By sending an Abort Session Request Message, the Controller End indicates that it does not expect the Responder End to maintain any test results for the aborted session.

7.4.8 Abort Session Response Message

The Abort Session Response Message is used by the Responder End for three purposes. These are:

- To acknowledge the Abort Session Request Message received from the Controller End
- To indicate that an error has occurred at the Responder End and the test session has stopped
- To indicate that SAT has been changed from enabled to disabled in the Responder End and the test session has been stopped.

Note: Test Session results are deleted when a Test Session is aborted.

7.4.9 Get Session Status Request Message

The Get Session Status Request Message is used by the Controller End to request the status of the test session. This is not used to retrieve test results while the test session is active, just to identify if the test session is in-progress or completed.

7.4.10 Get Session Status Response Message

The Get Session Status Response Message is used by the Responder End to respond to a Controller End Get Session Status Request Message. Possible responses are defined in section 10.

7.4.11 Fetch Session Results Request Message

The Fetch Session Results Request Message is used by the Controller End to request the results of a session from the Responder End. A Fetch Session Results Request is sent by the Controller End only after a session has stopped.

7.4.12 Fetch Session Result Response Message

The Fetch Session Results Response Message is used by the Responder End to acknowledge a Fetch Session Results Request Message received from the Controller End and to return the session results. If a Fetch Session Results Request Message was received while the test was active, the Fetch Session Results Response Message would be sent with a status code that rejects the request since no results are currently available.

7.4.13 Delete Session Request Message

The Delete Session Request Message is used by the Controller End to inform the Responder End that the session has completed and the Responder End can delete any results from the session and release any other resources related to the session.

7.4.14 Delete Session Response Message

The Delete Session Response Message is used by the Responder End to acknowledge a Delete Session Message received from the Controller End.

7.5 SAT Protocol Flow

Figure 10 provides an example of how SAT Control Protocol functions are used and how SAT Control Protocol Messages are exchanged between the Controller and Responder Ends of the SAT. Two Test sessions are shown running concurrently; one Forward session and one Backward session. A separate set of SAT Control Protocol Messages is exchanged for each Test Session.

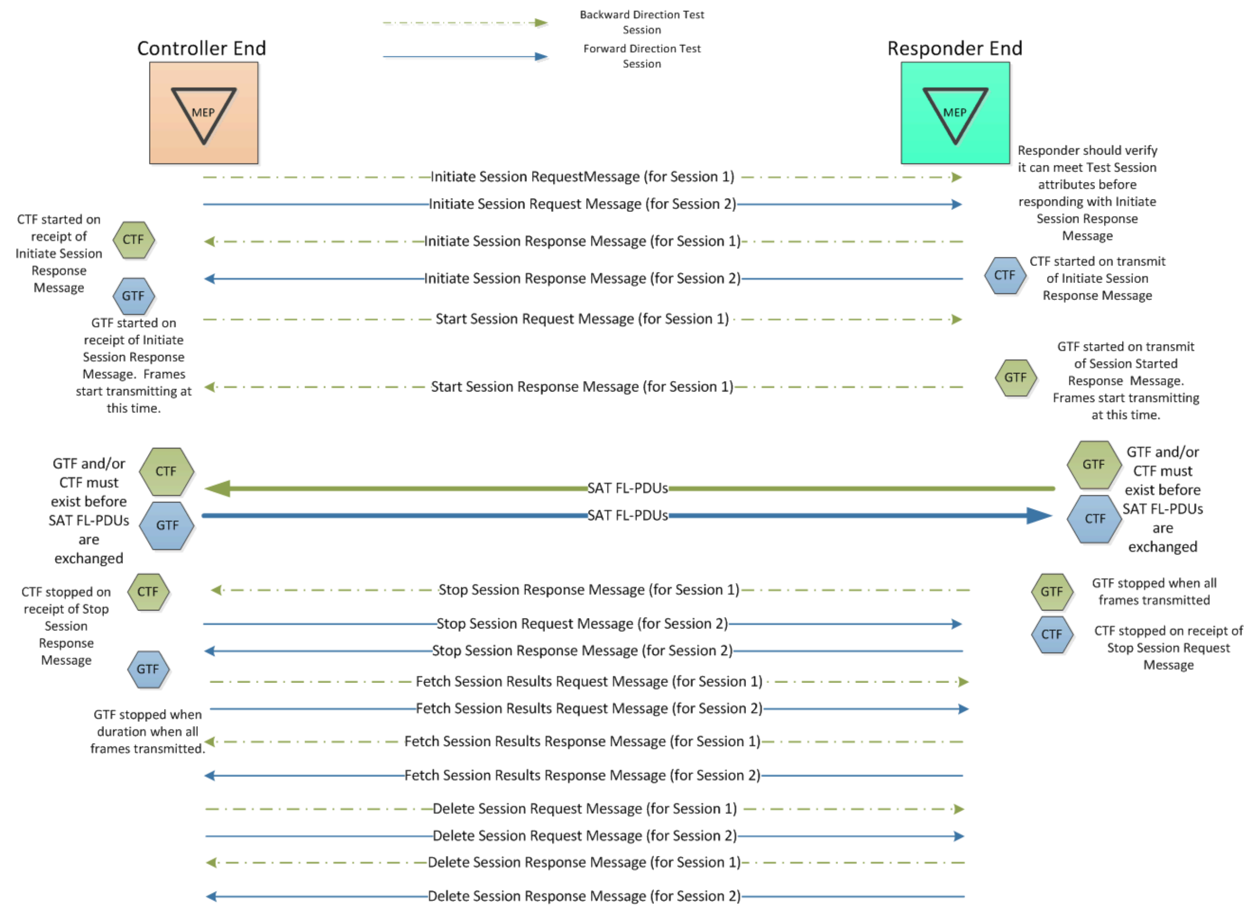

Backward Direction Test Session
Forward Direction Test Session
Controller End
Responder End
MEP
MEP
Responder should verify it can meet Test Session attributes before responding with Initiate Session Response Message
Initiate Session RequestMessage (for Session 1)
Initiate Session Request Message (for Session 2)
CTF started on receipt of Initiate Session Response Message
CTF
Initiate Session Response Message (for Session 1)
Initiate Session Response Message (for Session 2)
CTF
CTF started on transmit of Initiate Session Response Message
GTF
GTF started on receipt of Initiate Session Response Message. Frames start transmitting at this time.
Start Session Request Message (for Session 1)
GTF started on transmit of Session Started Response Message. Frames start transmitting at this time.
GTF
Start Session Response Message (for Session 1)
GTF and/or CTF must exist before SAT FL-PDUs are exchanged
CTF
GTF
SAT FL-PDUs
SAT FL-PDUs
GTF
CTF
GTF and/or CTF must exist before SAT FL-PDUs are exchanged
CTF stopped on receipt of Stop Session Response Message
CTF
Stop Session Response Message (for Session 1)
GTF
GTF stopped when all frames transmitted
Stop Session Request Message (for Session 2)
CTF
CTF stopped on receipt of Stop Session Request Message
Stop Session Response Message (for Session 2)
GTF
Fetch Session Results Request Message (for Session 1)
GTF stopped when duration when all frames transmitted.
Fetch Session Results Request Message (for Session 2)
Fetch Session Results Response Message (for Session 1)
Fetch Session Results Response Message (for Session 2)
Delete Session Request Message (for Session 1)
Delete Session Request Message (for Session 2)
Delete Session Response Message (for Session 1)
Delete Session Response Message (for Session 2)

Figure 10 SAT Control Protocol PDU Protocol Exchange Example

The Controller End of the SAT initiates the test session, defines the parameters for the test session and passes these parameters to the Responder End using the SAT Control Protocol Initiate Session Request Message.

The Responder End determines if it can meet the parameters of the test session and replies with an Initiate Session Response Message. The applicable Response Codes are used in the Initiate Session Response Message to indicate whether the Responder End can support the parameters of the test session. Once the test session parameters have been agreed to between the Controller and Responder Ends, the session can be started. If it is a Backward test, the Controller End sends a Start Session Request Message to start the test. The Responder End responds with the Session Started Message and the test begins. If it is a Forward test, then no Start Session Request message is needed and the Controller End simply starts the test.

The Generator and Collector now exchange SAT Test FL-PDUs if the test session is measuring frame loss. If frame delay measurements are required, the associated MEPs exchange SAT Test FD-PDUs. At the completion of the test session in the Forward direction, the Controller End sends a Stop Session Request Message to the Responder End. The Responder End responds with a Session Stopped Message and the test stops.

At the completion of the test session in the Backward direction, the Responder End sends a Stopped Session Response message to the Controller End.

The Generator and Collector stop exchanging SAT Test FL-PDUs and, if frame delay measurements were being performed, the Controller End stops sending FD-PDUs.

The Controller End uses the Fetch Session Results Request Message to request the test results from the Responder End. The Responder End responds with a Fetch Session Results Response Message to provide the results of the measurement to the Controller End. Session test results are contained within a TLV in the Fetch Session Results Response Message.

Upon receipt of the Fetch Session Results Response Message from the Responder End, the Controller End sends a Delete Session Request Message to the Responder End. This message is used to inform the Responder End that it can delete the session test results and release any resources associated to the test session.

To acknowledge the receipt of the Delete Session Request Message, the Responder End sends the Delete Session Response Message. The Controller End can now release any resources associated with the test session.

Figure 10 shows it is possible to have multiple test sessions between the same two end points running simultaneously. Each test session spawns an instance of the GTF and an instance of the

CTF, and can produce an autonomous set of session test results. This allows tests in each direction (Forward and Backward), or tests of different CoS where specified by SAT, to be performed simultaneously on a single OVC.

7.6 GTF and CTF Processing Requirements

The Controller End and Responder End exchange information in the Initiate Session Request Message and Initiate Session Response Message to enable the CTF and GTF to be set up correctly for the session. This information includes the MAC addresses of the CTF and GTF, which may be different from the MAC addresses of the Controller End MEP and the Responder End MEP.

For a Forward test session the Controller End provides the following information:

- The MAC address of the GTF
- The PCP/DEI value of green and yellow frames
- The Destination MAC address if the test is using Broadcast or Multicast frames
- The duration of the test

For a Forward test session the Responder End returns the MAC address of the CTF in the Initiate Session Response Message. The SAT FS that is used by the CTF at the Responder End to identify FL-PDUs is determined by the SAT FS of the Control Protocol Frame, i.e., only FL-PDUs belonging to the same SAT FS as the Control Protocol Frame are counted. The GTF at the Controller End uses the CTF MAC address as the destination MAC address of the FL-PDU (if a unicast frame is used in the test session). The CTF at the Responder End counts frames received that match the criteria provided by the Controller End.

For a Backward test session, the Controller End provides the following information:

- The MAC address of the CTF
- The PCP/DEI value of green and yellow frames
- The Destination MAC address if the test is using Broadcast or Multicast frames
- L2CP EtherType/SAP (if L2CP test)
- Frame length (if FL-PDU)
- Frame Pattern (PRBS31/8 octet pattern)
- Frame quantity and interval or
- Green/Yellow Information Rate and duration

For a Backward test session the Responder End returns the MAC address of the GTF in the Initiate Session Response Message. The VLAN ID(s) that are used by the GTF at the Responder End when generating FL-PDUs are determined by the SAT FS of the Control Protocol PDU Frame, i.e., FL-PDUs are generated such that they belong to the same SAT FS as the Control Protocol Frame. Similar to a Forward test session, the GTF at the Responder End uses the CTF MAC address as the destination MAC address of the FL-PDU (for a unicast test). The CTF at the Controller End counts and discards frames received that match the criteria.

The criteria used by a CTF for a given session to determine which frames to count are:

- Source MAC Address
- Destination MAC Address
- SAT FS
- PCP and DEI

For green frames, the CTF counter matches the green PCP value and DEI 0 (as the DEI is always 0 for green frames). If the test also includes yellow frames, then there is a separate counter for yellow frames which matches the same Source MAC, Destination MAC and SAT FS, and the yellow PCP value and yellow DEI value.

Note: the yellow PCP value equals the green PCP value if and only if the DEI = 1.

[R1] A CTF for a session **MUST** identify frames to count for the session based on the Source MAC, Destination MAC, SAT FS, PCP and DEI fields in the frame using the values specified for the session.

[R2] A CTF for a session **MUST** begin counting the matching frames (and corresponding octets) as soon as the session is created.

[R3] A CTF for a session **MUST** discard all matching frames for the session once they have been counted.

Note: the applicable Source MAC address, Destination MAC address, SAT FS, green PCP, yellow PCP and yellow DEI for a session are determined as described in section 10.

8. SAT Test PDU Formats

There are two SAT Test PDUs defined within this document. The FL-PDU is used for frame loss measurements and is defined in this document. The FD-PDU is used for frame delay measurements and is defined in ITU-T G.8013/Y.1731 [3]. The formats and requirements of these PDUs are defined below.

8.1 Frame Loss PDU

Frame loss measurement is performed using the FL-PDU, defined by this document. The FL-PDU includes Optional TLVs that are used to adjust the size of the Ethernet frame. The format of the FL-PDU, derived from the 1SL PDU defined in ITU-T Recommendation G.8013/Y.1731 [3], is shown in Figure 11. Modifications to the 1SL PDU were made since the FL-PDU is generated by the GTF and not a MEP. Some fields in the 1SL are not required for the purposes of the FL-PDU.

When consecutive octets are used to represent a binary number, the lower octet number has the most significant value. The bits in an octet are numbered from 1 to 8, where bit 1 is the least significant bit (LSB) and bit 8 is the most significant bit (MSB).

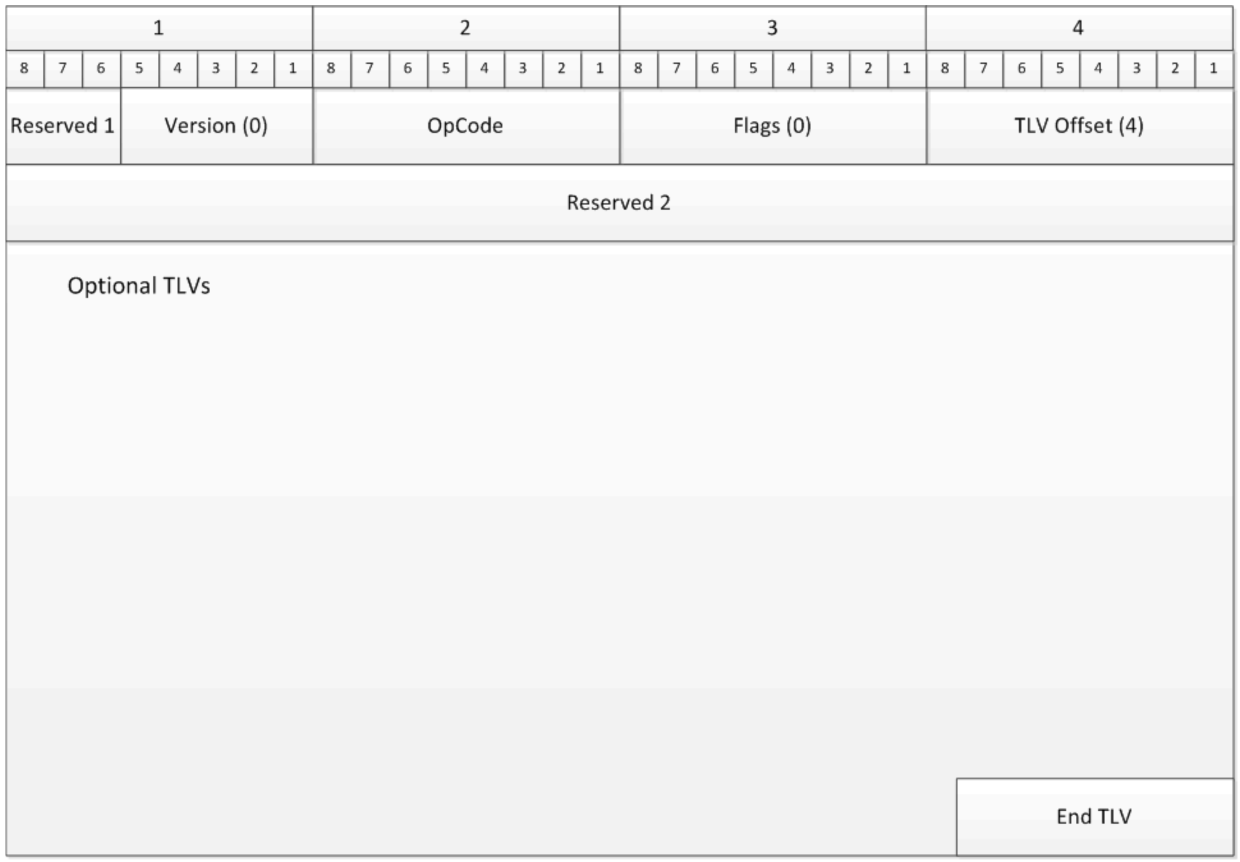


Figure 11 FL-PDU Format

[R4] Frame loss measurement for SAT **MUST** use the FL-PDU as shown in Figure 11.

The FL-PDU is inserted into an Ethernet Frame using the OUI Extended EtherType defined in IEEE 802 [17] i.e., with an EtherType of 0x88b7 with the MEF assigned OUI of 90-FF-79 and subtype of 01 so that the frames are identified as SAT traffic. The SAT traffic is treated the same as customer traffic for test purposes but is uniquely identified by the EtherType, OUI and sub-type. Figure 12 shows the FL-PDU inserted into an Ethernet frame. For details on the FL-PDU see Figure 11.

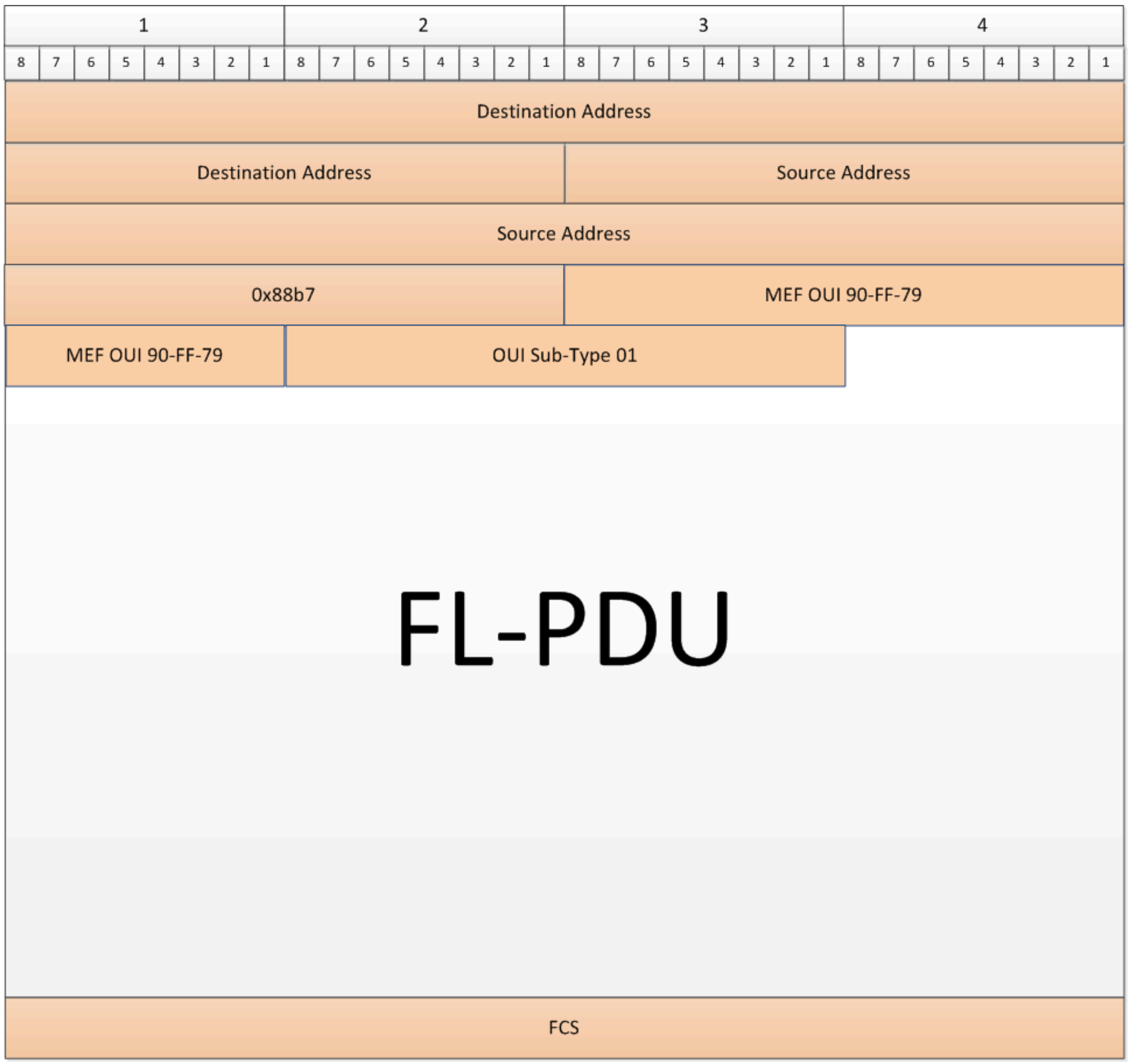


Figure 12 FL-PDU in an Ethernet Frame

The Ethernet frame can contain 0 or 1 S-Tag, and/or 0 or 1 802.1Q C-Tag, between the Source Address and the EtherType. The example above shows 0 tags. The methods used to determine the Source and Destination MAC addresses are described in section 10.4.2.2.

8.1.1 Reserved 1

The Reserved 1 field is reserved for future use by MEF.

[R5] The Reserved 1 field **MUST** be set to 0 on transmit and ignored on receive.

8.1.2 Version

The Version field is used to identify the version of FL-PDU. This is set to 0.

[R6] The Version field **MUST** be set to 0 for PDUs following this specification.

[R7] A received FL-PDU **MUST** be processed and validated as described in G.8013/Y.1731 [3] clauses 11.2 and 11.3.

8.1.3 OpCode

The OpCode field is set to 1 to indicate that this is a FL-PDU.

[R8] On transmit the OpCode field **MUST** be set to 1 by the GTF.

[R9] On receive, FL-PDUs with OpCode field values other than 1 **MUST** be discarded.

8.1.4 Flags

[R10] The Flags field is unused and is set to 0.

[R11] On transmit the Flags field **MUST** be set to 0 by the GTF.

[R12] On receive the Flags field **MUST** be ignored by the CTF.

8.1.5 TLV Offset

The TLV Offset indicates the number of octets from this field at which the TLV fields, if any, begin.

[R13] The TLV Offset **MUST** be set to indicate the start of the first TLV.

[R14] The transmitted TLV Offset **MUST** be 4.

[R15] A valid received FL-PDU **MUST** have a TLV Offset of 4 or greater.

[R16] The TLV Offset **MUST** be set to a value that indicates that the TLVs begin at or before the start of the FCS field within the frame containing the PDU.

Note: If the TLV offset indicates that the TLVs start at the start of the FCS field, then there are no TLVs present in the FL-PDU.

8.1.6 Reserved 2

These four octets are reserved for future use (possible use could be as a sequence number).

[R17] On transmit the Reserved 2 field **MUST** be set to 0 by the GTF.

[R18] On receive the Reserved 2 field **MUST** be ignored by the CTF.

Note: The implementation of a Sequence Number at line rate is thought to be difficult to implement at higher speeds and therefore is not included in this revision of the FL-PDU definition.

8.1.7 Optional TLVs

FL-PDUs may contain Optional TLVs. The general format of an Optional TLV is shown in the figure below:

Type (1 octet)	Length (2 octets)	Value (0 – 65535 octets)

Figure 13 FL-PDU TLV Format

These fields are described further below.

[R19] The first TLV in the FL-PDU, if any, **MUST** start at the offset identified by the First TLV Offset field in the FL-PDU.

[R20] Each subsequent TLV **MUST** start immediately after the end of the previous TLV.

8.1.7.1 TLV Type

The Type field indicates the type of the TLV. If the value is 0, there are no Length or Value fields. The table below indicates the valid types:

Type	Meaning
0	End TLV
3	Data TLV
1-2, 4-255	Reserved

Table 3 SAT FL-PDU Types

[R21] A TLV in an FL-PDU transmitted by a GTF **MUST** have a Type value that is not Reserved.

[R22] If an FL-PDU received by a CTF contains a TLV with a Reserved type, the TLV **MUST** be ignored.

[R23] A FL-PDU that is ignored **MUST** still be counted by the CTF.

[R24] If the end of the FL-PDU (including any TLVs) is not immediately followed by the first octet of the FCS, the FL-PDU **MUST** contain an End TLV.

[R25] If the FL-PDU contains an End TLV, it **MUST** be the last TLV in the FL-PDU.

8.1.7.2 TLV Length

The Length field indicates the length, in octets, of the Value field. If the Length is 0, there is no Value field.

[R26] The Length in an FL-PDU **MUST NOT** indicate that the TLV Value extends beyond the end of the Ethernet Frame data.

8.1.7.3 TLV Value

The Value field contains data that depends on the Type.

8.1.7.4 Data TLV

The Data TLV is used to ensure that the Ethernet frame has the length selected for the test instance. This is shown in Figure 11, as the Optional TLVs. The start of the Data TLV is indicated by the TLV offset value.

[R27] The Data TLV **MUST** be used to meet the Ethernet frame length at the GTF for the Test Session

[R28] The Data TLV **MUST** contain either a repeating eight octet pattern or a PRBS31 pattern in the TLV Value field as a part of the test FL-PDU.

[R29] A GTF **MUST** support generating an eight octet pattern.

[R30] The repeating eight octet pattern **MUST** restart in each PDU.

[O1] A GTF **MAY** support generating a Pseudo-Random Bit Sequence (PRBS31) pattern as defined in ITU-T O.150 instead of a repeating eight octet pattern as a part of the test FL-PDU.

[O2] An ETE implementation supporting multiple GTFs **MAY** allow a single PRBS generator to be used for all GTFs.

[R31] An FL-PDU **MUST NOT** contain more than one Data TLV.

The PRBS31 pattern is used to ensure that there is no compression of the frame containing the FL-PDU which would impact any bandwidth measurements. It is not used to detect any errors within the frame. A user has the ability to specify the use of the repeating eight octet pattern if that is desired. The eight octet pattern might be used to isolate the cause of a pattern specific problem.

8.2 FD-PDU

Frame Delay measurements are performed using a Delay Measurement Message version 1 (DMMv1) as defined in ITU-T Recommendation G.8013/Y.1731 [3].

The MEP at the Controller End transmits a DMMv1 PDU. The MEP at the Responder End of the EVC or OVC responds with a DMR per normal SOAM operations. The frame delay calculation is performed at the Controller End of the Test Session.

[R32] To perform Frame Delay measurements, the MEP associated with the Controller End **MUST** support the requirements for single ended frame delay measurement defined in MEF 35 [10] and ITU-T G.8013/Y.1731 [3].

[R33] The Controller End **MUST** have a method of correlating the Delay measurements to the Test Session ID.

[R34] The MEP associated with the Responder End **MUST** support the requirements for single ended frame delay measurement defined in R54 of MEF 35 [10].

8.2.1 Interleaving FL-PDU and FD-PDU

When both Frame Loss and Frame Delay measurements are required, two different PDUs, the FL-PDU and the FD-PDU, are used to perform testing and are interleaved in the service being tested. While using the FD-PDU solution can reduce the complexity of some devices, it does not come without a cost. Since Frame Delay is not measured on every frame transmitted, changes in latency that last less than the FD-PDU interval may not be detected. It is recommended that the FD-PDU interval selected be short enough to ensure the majority of the variations in latency are detected. Operators need to account for the possibility of additional bandwidth due to interleaving FL-PDUs and FD-PDUs being sent simultaneously.

Note 1: FD-PDUs are not considered for Loss or Throughput measurements.

Note 2: The selection of the FD-PDU period is an ETSS function and is outside the scope of this document.

9. SAT Control Protocol State Machines

This section defines the State Machines for the SAT Control Protocol. These State Machines are instantiated in MEPs within ETEs at the Controller and Responder Ends. The state machines defined are:

- Controller End Global State Machine
- Controller End Session State Machine
- Responder End Global State Machine
- Responder End Session State Machine

State machines have been defined for both the Controller and Responder Ends and handle both Forward and Backward tests. Implementations of the SAT Control Protocol do not have to implement these state machines as written, but implementations do have to provide the same externally visible behaviors. This allows flexibility in implementations while still maintaining interoperability.

[R35] MEPs instantiating the SAT Control Protocol state machines **MUST** perform MEG Level filtering on received SCM/SCR messages as defined in 802.1Q [1], G.8013 [3]and G.8021 [15].

[R36] An implementation of a MEP supporting SAT Controller End functionality **MUST** provide externally visible behavior indistinguishable from that described by the state machines described in section 9.2.

[R37] An implementation of a MEP supporting SAT Responder End functionality **MUST** provide externally visible behavior indistinguishable from that described by the state machines described in section 9.3.

[R38] An implementation of a MEP supporting SAT Controller End functionality **MUST** allocate Test Session IDs that are unique over all sessions initiated at the MEP, or at any other MEP that shares the same MAC address.

Note: Multiple MEPs may share the same MAC address if they are at different levels or different VLANs on the same interface, or if they are Up MEPs on different interfaces and the Shared MP address model is used as described in IEEE 802.1Q [1] section J.6.

[R39] An implementation of a MEP supporting SAT Controller End or Responder End functionality **MUST NOT** accept a new session that has the same source MAC, destination MAC and SAT FS, and has the same green PCP or the same yellow PCP and yellow DEI (if defined), as an existing session. This applies whether it is the Controller End or the Responder End for the existing session.

9.1 State Machine Descriptions

Each state machine is described as follows:

- List of events handled by the state machine (i.e., inputs)

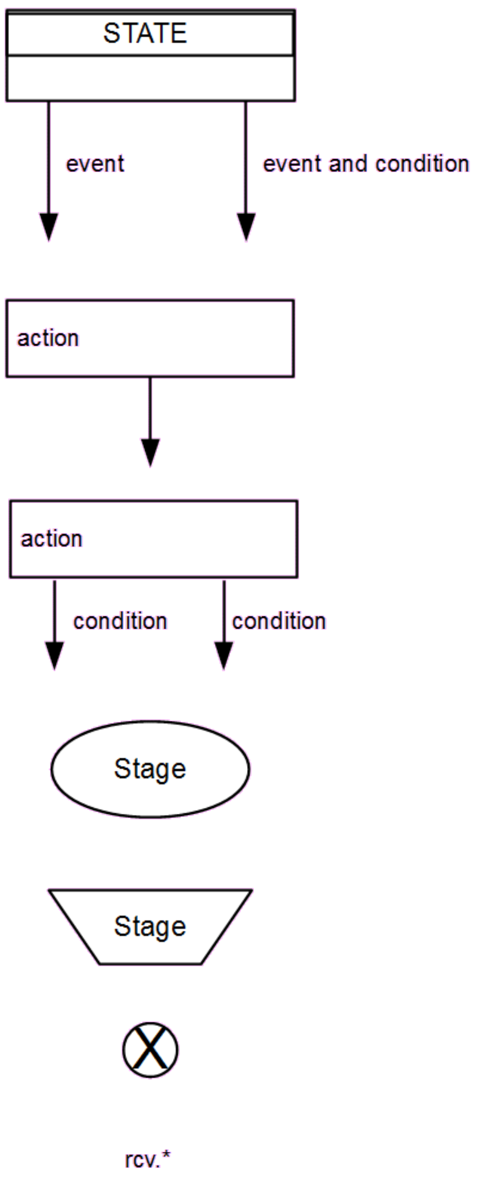
- List of actions taken by the state machine (i.e., outputs)
- List of functions used by the state machine (if any)
- Diagrammatic representation of the state machine

The events and actions are assigned two-part names, where the first part indicates the type of event or action (e.g., relating to the ETSS, to sent/received PDUs, to timers, etc.). Other than this the names have no significance. Some events and actions are described with parameters; in the case of events this indicates information that is input into the state machine along with the event; in the case of actions it indicates information that is output from the state machine when the action is triggered. In both cases, the parameters and their possible values are described along with the event or action.

Some state machines make use of "functions" these are used simply as a shorthand to describe auxiliary behavior without cluttering the state machine diagram itself.

For ease of description, some State Machine diagrams are split into "stages". This is simply because the complete state machine is too large to fit on a single page; the split into stages has no impact on the behavior that is described by the state machine. The stages are linked in the diagrams by a "Continue at Stage X" symbol and a "Start of Stage X" symbol.

The state machine behavior itself is described diagrammatically, using the conventions shown in Figure 14.



State Machine state

Events that can occur in the state, optionally with conditions. Arrow shows the following Action or State

Execute the specified actions

Arrow shows the following Action or State (unconditional)

Arrow shows the following Action or State (with conditions)

Continue at the given Stage of the state machine.

Start of the given Stage of the state machine.

Termination of the State Machine

Abbreviation for multiple network events. The list of events is given separately below the figure.

Note: other messages than those listed are not part of * and are ignored.

Figure 14 State Machine Figure Conventions

9.2 Controller End State Machines

A Controller End MEP contains two State Machines:

- Controller End Global State Machine (1 instance)
- Controller End Session State Machine (1 instance per active session)

For ease of description, the Controller End Session State Machine is split into several stages:

- Create
- Pending
- Start
- Run
- Stop
- Fetch
- Delete
- Abort
- Status

Most of these stages run consecutively; the exception is the "Status" stage, which runs concurrently with all of the other stages.

At a Controller End MEP, a session (and hence an instance of the Controller End Session State Machine) is identified uniquely by the locally assigned session ID.

An overview of the Controller End State Machines is shown in Figure 15.

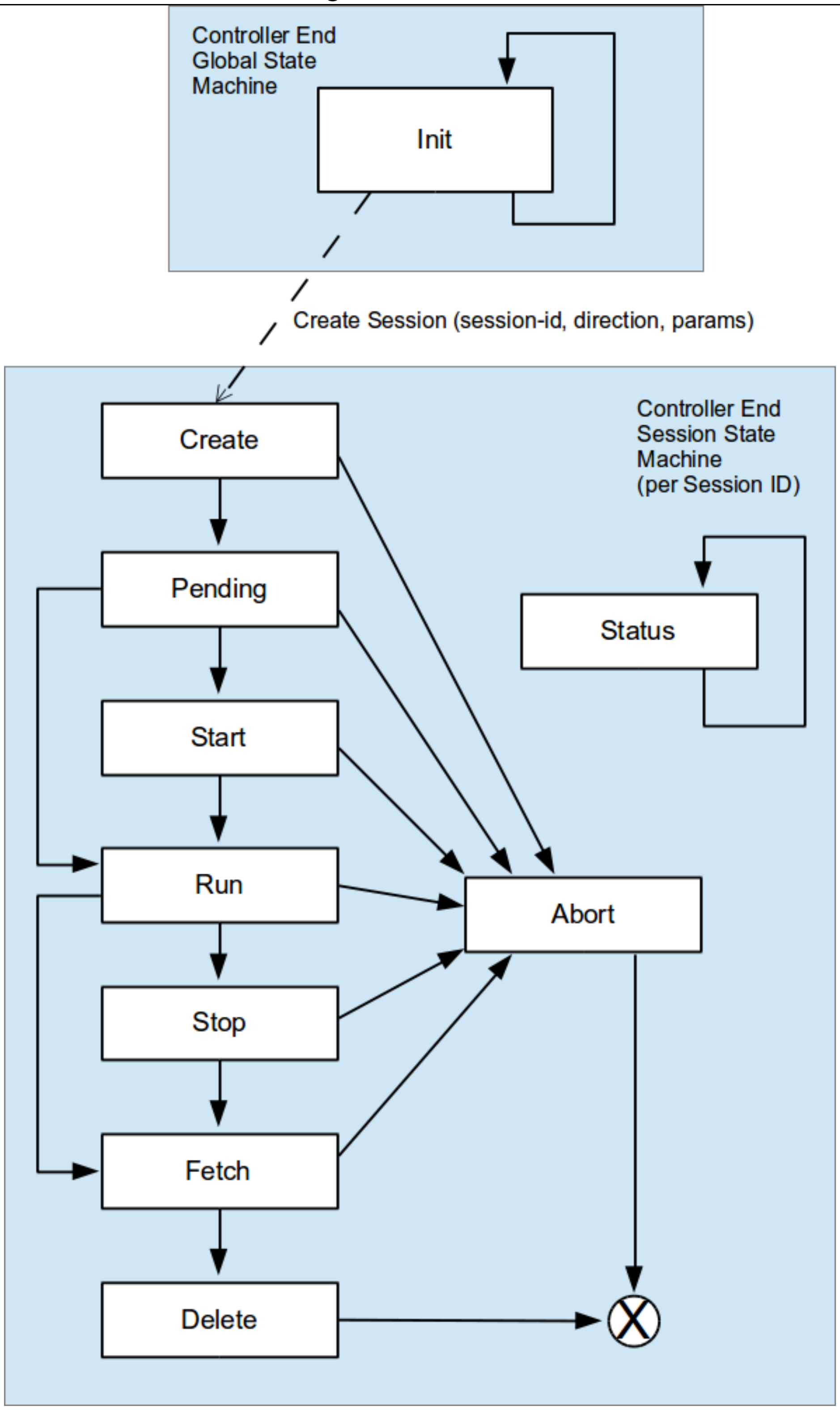


Figure 15 Controller End State Machines Overview

9.2.1 Controller End Global State Machine

There is a single instance of the Controller End Global State Machine in each MEP supporting SAT Controller End functionality. This is described in the following Sections.

9.2.1.1 Controller End Global State Machine Events

The Controller End Global State Machine handles the following events:

Manageability events:

- `etss.create_session(direction, params)`: A command has been received from the ETSS to create a new session, with the given session parameters. The `direction` is "FORWARD" (i.e., Controller End to Responder End) or "BACKWARD" (i.e., Responder End to Controller End). The `params` include all the details needed to create the session, including the MAC address of the Responder MEP and the information about the desired test traffic, such as the SAT FS, the PCP, the number of frames to send, the interval between frames, the frame length, etc.

Network events:

- `rcv.unknown()`: A valid SAT PDU response message has been received (that is, an SCR conforming to the specification in section 10.3) for a session ID for which the Controller End MEP has no corresponding active Session State Machine.

9.2.1.2 Controller End Global State Machine Actions

The Controller End Global State Machine can take the following actions:

Manageability actions:

- `etss.create_sn_error(error)`: The ETSS is notified of the specified `error` having occurred when attempting to create the session. Possible errors are:
    - LOCAL_REJECT: The Controller End cannot create the session locally, e.g., because the parameters are not supported or the device is out of resources.
- `etss.create_sn_success(session_id)`: The ETSS is notified that the session has been successfully created at the Controller End, and that the specified `session_id` has been allocated. Note that this does not indicate that the session has been successfully created at the Responder End.

Other actions:

- `spawn.session(session_id, direction, params)`: Spawn a Controller End Session State Machine for the given `session_id`, `direction` and `params`.

9.2.1.3 Controller End Global State Machine Functions

The Controller End Global State Machine uses the following functions:

- `check_existing_sessions(params):` This function checks whether there are existing sessions that could conflict with the new session. It is used only for Forward tests. If the test is for a broadcast or multicast destination MAC, and if among all the MEPs that share the same GTF (and hence the same source MAC for FL-PDUs) there is an existing session for the same destination MAC address and SAT FS, and the same green PCP, or the same yellow PCP (if defined), and yellow DEI (if defined), this function returns FALSE; otherwise it returns TRUE.

Note: For a unicast test, this check cannot be performed until the CTF MAC Address is received in the Initiate Session Response message; this is handled by the Controller End Session State Machine.

- `check_session(direction, params):` This function checks to see if a session with the requested direction and parameters can be supported at the Controller End. It returns a Boolean indicating TRUE if the session can be supported or FALSE otherwise. The specific checks are implementation-specific, for example, to verify there are sufficient resources available to create the session.

- `allocate_session_id():` This function allocates a Controller End `session_id` for the session that is different to the session ID for any existing Controller End Session State Machine in the MEP, or in any other MEPs that share the same MAC address.
  Note that the exact method by which a unique session ID is allocated is implementation specific. The session ID is required to be unique among all active sessions within the Controller End MEP (and any other MEPs that share the same MAC address); however it could be implemented, for example, such that it is unique over all active sessions with the NE. It is recommended not to re-allocate session IDs immediately when a session for a given session ID terminates.

9.2.1.4 Controller End Global State Machine

The Controller End Global State Machine is shown in Figure 16.

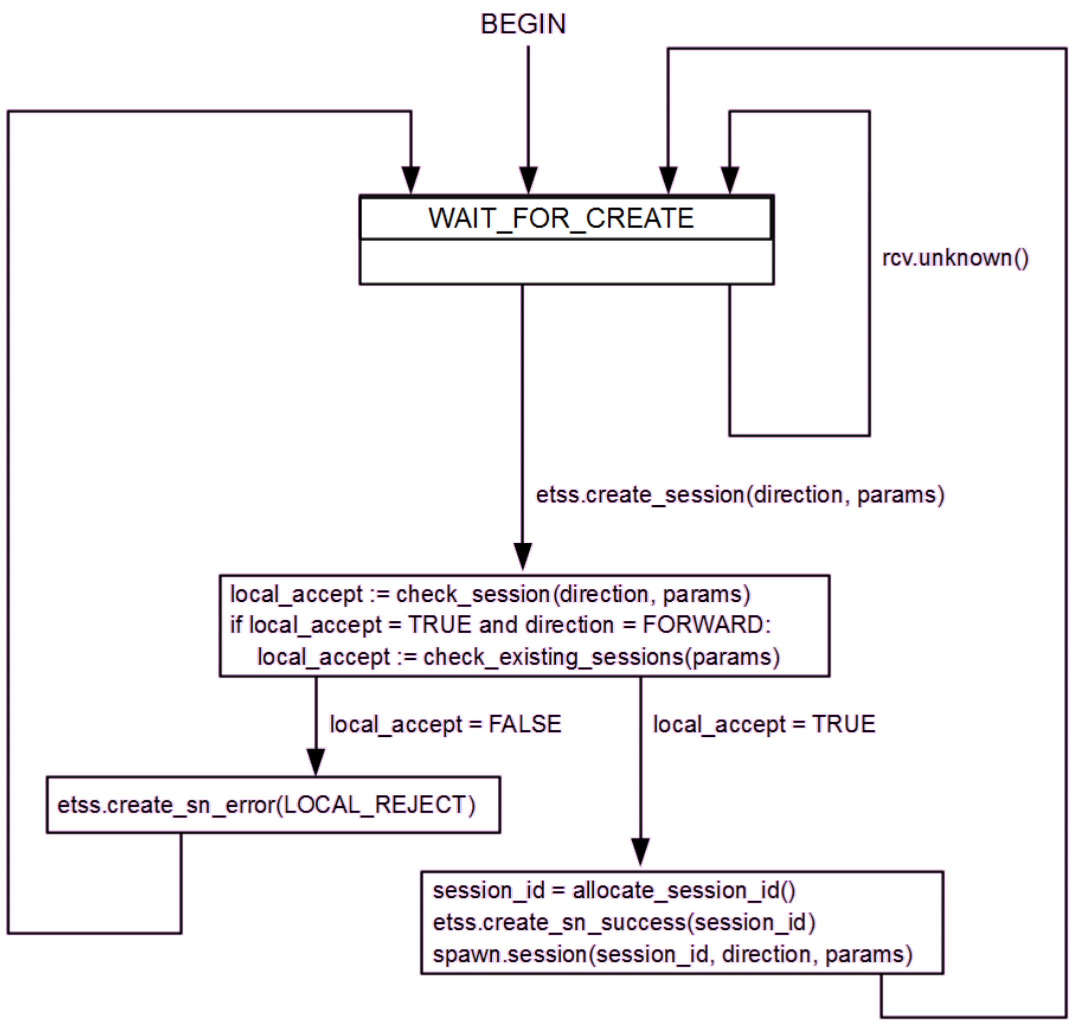

BEGIN
WAIT_FOR_CREATE
rcv.unknown()
etss.create_session(direction, params)
local_accept := check_session(direction, params)
if local_accept = TRUE and direction = FORWARD:
local_accept := check_existing_sessions(params)
local_accept = FALSE
local_accept = TRUE
etss.create_sn_error(LOCAL_REJECT)
session_id = allocate_session_id()
etss.create_sn_success(session_id)
spawn.session(session_id, direction, params)

Figure 16 Controller End Global State Machine

Note: Receiving an `etss.create_session()` event always results in exactly one of `etss.create_sn_error()` or `etss.create_sn_success()`; that is, when the ETSS attempts to create a session, it always receives back exactly one response: i.e., either an error, or the allocated session ID. The mechanism by which the ETSS interacts with the Controller End MEP is out of scope of this specification; however, it is assumed that the mechanism is such that the ETSS can correctly match the response to its original request. Subsequent interactions regarding the session (if it was successfully created) are always in the context of the allocated session ID, as described in section 9.2.2.

9.2.2 Controller End Session State Machine

There is an instance of the Controller End Session State Machine in the Controller End MEP for each session that has been created by the Controller End Global State Machine in that MEP. An instance of the Session State Machine is associated with exactly one locally-allocated session ID. The instance continues running until it terminates according to the specification below: after it has terminated, the corresponding session ID is no longer active on the MEP and may be reused for a new session.

The instance also has an associated direction (FORWARD or BACKWARD) and session parameters, as specified when the instance is spawned. The CTF (for a Backward session) or GTF (for a Forward session) is created at the same time as the state machine is spawned, and is started as shown in the state machine diagrams. Similarly, the GTF or CTF is deleted and any resources released when the state machine terminates.

Assuming the test completes successfully, by the time the state machine terminates, the Controller End MEP will have both its local results and the results it has fetched from the Responder End. The results data is not deleted when the state machine terminates. It is passed back to the ETSS, although the mechanism by which this is done is out of scope of this document. For example, it could be pushed immediately to the ETSS via a notification, or it could be stored in the NE or ETE until the ETSS requests it.

All events and actions for the Controller End Session State Machine instance are specific to the MEP and session ID for which the state machine instance is running. Note that the MEP and session ID are not shown explicitly as parameters to the State Machine events and actions, but may be presumed implicitly. In addition, the state machine instance uses the session-id, direction and params that were specified when it was spawned.

9.2.2.1 Controller End Session State Machine Events

The Controller End Session State Machine handles the following events:

Manageability events:

- `etss.start_session()`: A request has been received from the ETSS to start the session. This event can only occur in READY state; any attempt by the ETSS to start a session when in any other state results in an error.
- `etss.abort()`: A request has been received from the ETSS to abort the session.
- `etss.status_rq()`: A request has been received from the ETSS for the session status.

Network events:

- `rcv.init_sn_rsp_ok(mac)`: A valid Initiate Session Response Message (that is, one conforming to the specification in section 10.3) has been received for this session, with the Response code set to 0 (NO_ERROR). If the message contained a MAC Address TLV, `mac` is the value of the TLV; otherwise it is unspecified.
- `rcv.init_sn_rsp_error(error)`: A valid Initiate Session Response Message (that is, one conforming to the specification in section 10.3) has been received for this session, with the Response code `error` set to one of the values in Table 7 other than 0 (NO_ERROR).
- `rcv.sn_started_ok()`: A valid Start Session Response Message (that is, one conforming to the specification in section 10.3) has been received for this session, with the Response code set to 0 (NO_ERROR).
- `rcv.sn_started_error(error)`: A valid Start Session Response Message (that is, one conforming to the specification in section 10.3) has been received for this session, with the Response code `error` set to one of the values in Table 7 other than 0 (NO_ERROR).
- `rcv.sn_stopped_ok()`: A valid Stop Session Response Message (that is, one conforming to the specification in section 10.3) has been received for this session, with the Response code set to 0 (NO_ERROR).
- `rcv.sn_stopped_error(error)`: A valid Stop Session Response Message (that is, one conforming to the specification in section 10.3) has been received for this session, with the Response code `error` set to one of the values in Table 7 other than 0 (NO_ERROR).
- `rcv.sn_status_ok(status)`: A valid Get Session Status Response Message (that is, one conforming to the specification in section 10.3) has been received for this session, with the Response code set to 0 (NO_ERROR), and containing the Responder End status value `status`.
- `rcv.sn_status_error(error)`: A valid Get Session Status Response Message (that is, one conforming to the specification in section 10.3) has been received for this session,

with the Response code `error` set to one of the values in Table 7 other than 0 (NO_ERROR).

- `rcv.sn_results(results)`: A valid Fetch Session Results Response Message (that is, one conforming to the specification in section 10.3) has been received for this session, containing the Responder End results value `results`.
- `rcv.sn_deleted_ok()`: A valid Delete Session Response Message (that is, one conforming to the specification in section 10.3) has been received for this session, with the Response code set to 0 (NO_ERROR).
- `rcv.sn_aborted_ok()`: A valid Abort Session Response Message (that is, one conforming to the specification in section 10.3) has been received for this session, with the Response code set to 0 (NO_ERROR).
- `rcv.sn_aborted_error(error)`: A valid Abort Session Response Message (that is, one conforming to the specification in section 10.3) has been received for this session, with the Response code `error` set to one of the values in Table 7 other than 0 (NO_ERROR).

Note: A Fetch Session Results Response Message or a Delete Session Response Message with a Response code set to a value other than 0 (NO_ERROR) is not valid (that is, does not conform to the specification in section 9.3). Therefore, no corresponding events are defined in the state machine.

Other events:

- `wait_timer.exp()`: The Wait Timer has expired
- `status_wait_timer.exp()`: The Status Wait Timer has expired
- `gtf.complete()`: The GTF has finished sending all the frames for the session.

9.2.2.2 Controller End Session State Machine Actions

The Controller End Session State Machine can take the following actions:

Manageability actions:

- `etss.error(state, error)`: The ETSS is notified of the specified `error` having occurred, when the state machine was in the specified `state`. The `state` is one of INITIATING, READY, STARTING, RUNNING, STOPPING, FETCHING_RESULTS, DELETING, ABORTING, or GETTING_STATUS. Possible errors include:
    - o REMOTE_ERROR(`remote_error`): An SCR was received containing a Response code set to `remote_error`.

  - NO_RESPONSE: An SCR was expected from the Responder End but none was received within the wait time.
  - NO_START: No start command was received from the ETSS within the wait time.
  - NOT_READY: A request was received from the ETSS to start the session, but the session is not yet ready, i.e., has not yet been successfully created at the Responder End.
  - ALREADY_STARTED: A request was received from the ETSS to start the session, but it has already been started by a previous request.
  - SESSION_CONFLICT: There is an existing session with the same source and destination MACs, SAT FS and PCP.

- `etss.info(event)`: The ETSS is notified of an `event` in the state machine; events include:
  - SN_CREATED – the session has been successfully created at the Responder End and is ready to be started.
  - SN_STARTED – the session has been started.
  - SN_STOPPED – the session has stopped at the end of the test.
  - SN_DELETED – the session has been deleted after completing successfully.
  - SN_ABORTED – the session has been deleted after having been aborted, either by the local ETSS or by the Responder End.

- `etss.status(state, status)`: The session's Controller End `state` and Responder End `status` is returned to the ETSS. The `state` is one of INITIATING, READY, STARTING, RUNNING, STOPPING, FETCHING_RESULTS, DELETING, or ABORTING; the `status` is one of the values from Table 12, or the special value NO_RESPONSE.

Network actions:

- `send.init_sn_rq(direction, params)`: Send an Initiate Session Request Message containing the session `direction` and `params`.
- `send.start_sn_rq()`: Send a Start Session Request Message
- `send.stop_sn_rq()`: Send a Stop Session Request Message
- `send.sn_status_rq()`: Send a Get Session Status Request Message

- `send.sn_results_rq()`: Send a Fetch Session Results Request Message
- `send.sn_delete_rq()`: Send a Delete Session Request Message
- `send.sn_abort_rq()`: Send an Abort Session Request Message

Other actions:

- `wait_timer.start(duration)`: Start the wait timer for the specified `duration.`
- `wait_timer.stop()`: Stop the wait timer.
- `status_wait_timer.start(duration)`: Start the status wait timer for the specified `duration.`
- `status_wait_timer.stop()`: Stop the status wait timer.
- `ctf.start(smac, params)`: Start the CTF, for the given source MAC address and for the SAT FS and PCP in the given parameters.

  Note: The destination MAC for the session is the CTF's MAC address.
- `gtf.start (dmac, params):` Start the GTF, for the SAT FS and PCP in the given parameters. If the parameters specify a broadcast or multicast test, the destination address is taken from the parameters; otherwise the given `dmac` is used. The source MAC address is the GTF's MAC address.
- `gtf.stop():` Stop the GTF, i.e., do not transmit any further frames.

9.2.2.3 Controller End Session State Machine Functions

The Controller End Session State Machine uses the following functions:

- `check_existing_sessions(dmac, params):` This function checks whether there are existing sessions that could conflict with the new session. It is only used for Forward tests. If the test is a unicast test, and if among all the MEPs that share the same GTF (and hence the same source MAC for FL-PDUs) there is an existing session for the same destination MAC address and SAT FS, and the same green PCP or the same yellow PCP and yellow DEI (if defined), this function returns FALSE; otherwise it returns TRUE.

Note: For a broadcast or multicast test, this check is performed by the Controller End Global State Machine before the session is created. However, this is not possible for unicast tests as the destination MAC address is not known until the Session Initiate Response message is received.

9.2.2.4 Controller End Session State Machine

The Controller End Session State Machine is shown in Figure 17, Figure 18, Figure 19, Figure 20, Figure 21, Figure 22, Figure 23, Figure 24, and Figure 25.

Note that the Status stage shown in Figure 25 runs in parallel with the other stages. It handles the `etss.status_rq()`, `rcv.sn_status_ok(status)`, `rcv.sn_status_error(error)`, and `status_wait_timer.exp()` events.

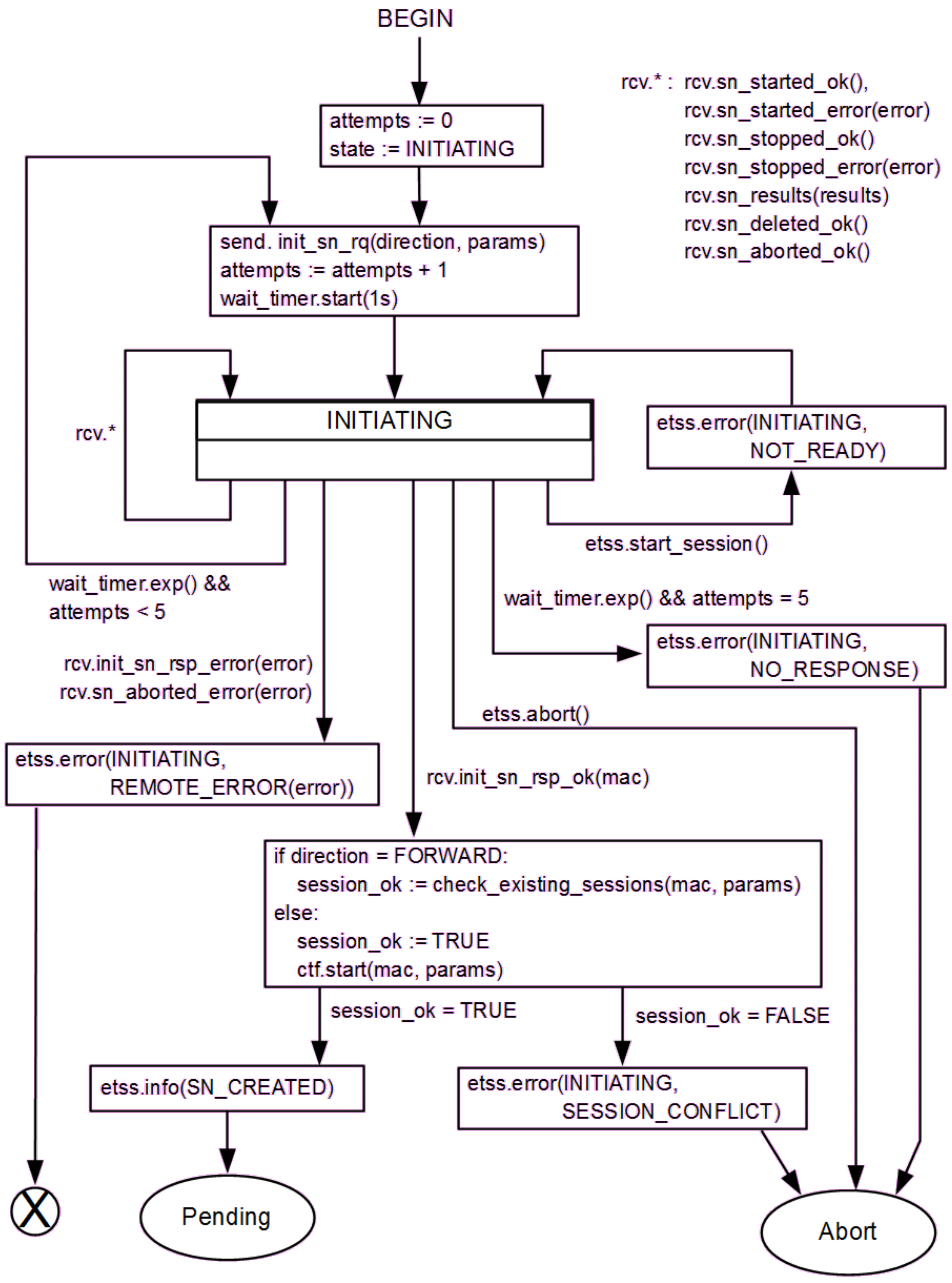

BEGIN
attempts := 0
state := INITIATING
rcv.* : rcv.sn_started_ok(),
rcv.sn_started_error(error)
rcv.sn_stopped_ok()
rcv.sn_stopped_error(error)
rcv.sn_results(results)
rcv.sn_deleted_ok()
rcv.sn_aborted_ok()
send. init_sn_rq(direction, params)
attempts := attempts + 1
wait_timer.start(1s)
rcv.*
INITIATING
etss.error(INITIATING, NOT_READY)
etss.start_session()
wait_timer.exp() && attempts < 5
wait_timer.exp() && attempts = 5
etss.error(INITIATING, NO_RESPONSE)
rcv.init_sn_rsp_error(error)
rcv.sn_aborted_error(error)
etss.abort()
etss.error(INITIATING, REMOTE_ERROR(error))
rcv.init_sn_rsp_ok(mac)
if direction = FORWARD:
session_ok := check_existing_sessions(mac, params)
else:
session_ok := TRUE
ctf.start(mac, params)
session_ok = TRUE
session_ok = FALSE
etss.info(SN_CREATED)
etss.error(INITIATING, SESSION_CONFLICT)
Pending
Abort

Figure 17 Controller End Session State Machine – Create Stage

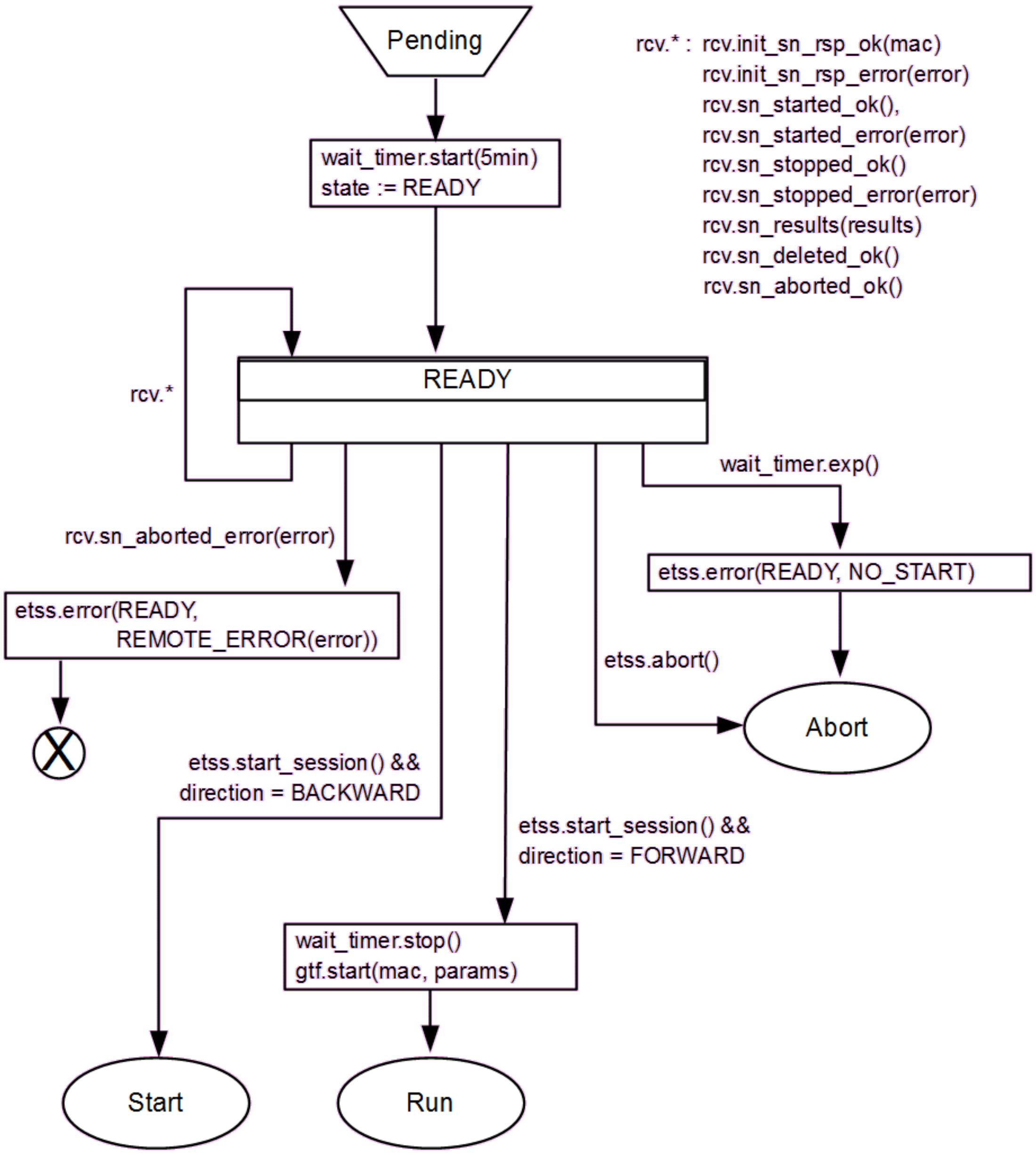

Pending
wait_timer.start(5min)
state := READY
rcv.* : rcv.init_sn_rsp_ok(mac)
rcv.init_sn_rsp_error(error)
rcv.sn_started_ok(),
rcv.sn_started_error(error)
rcv.sn_stopped_ok()
rcv.sn_stopped_error(error)
rcv.sn_results(results)
rcv.sn_deleted_ok()
rcv.sn_aborted_ok()
rcv.*
READY
wait_timer.exp()
rcv.sn_aborted_error(error)
etss.error(READY, NO_START)
etss.error(READY, REMOTE_ERROR(error))
etss.abort()
Abort
etss.start_session() && direction = BACKWARD
etss.start_session() && direction = FORWARD
wait_timer.stop()
gtf.start(mac, params)
Start
Run

Figure 18 Controller End Session State Machine – Pending Stage

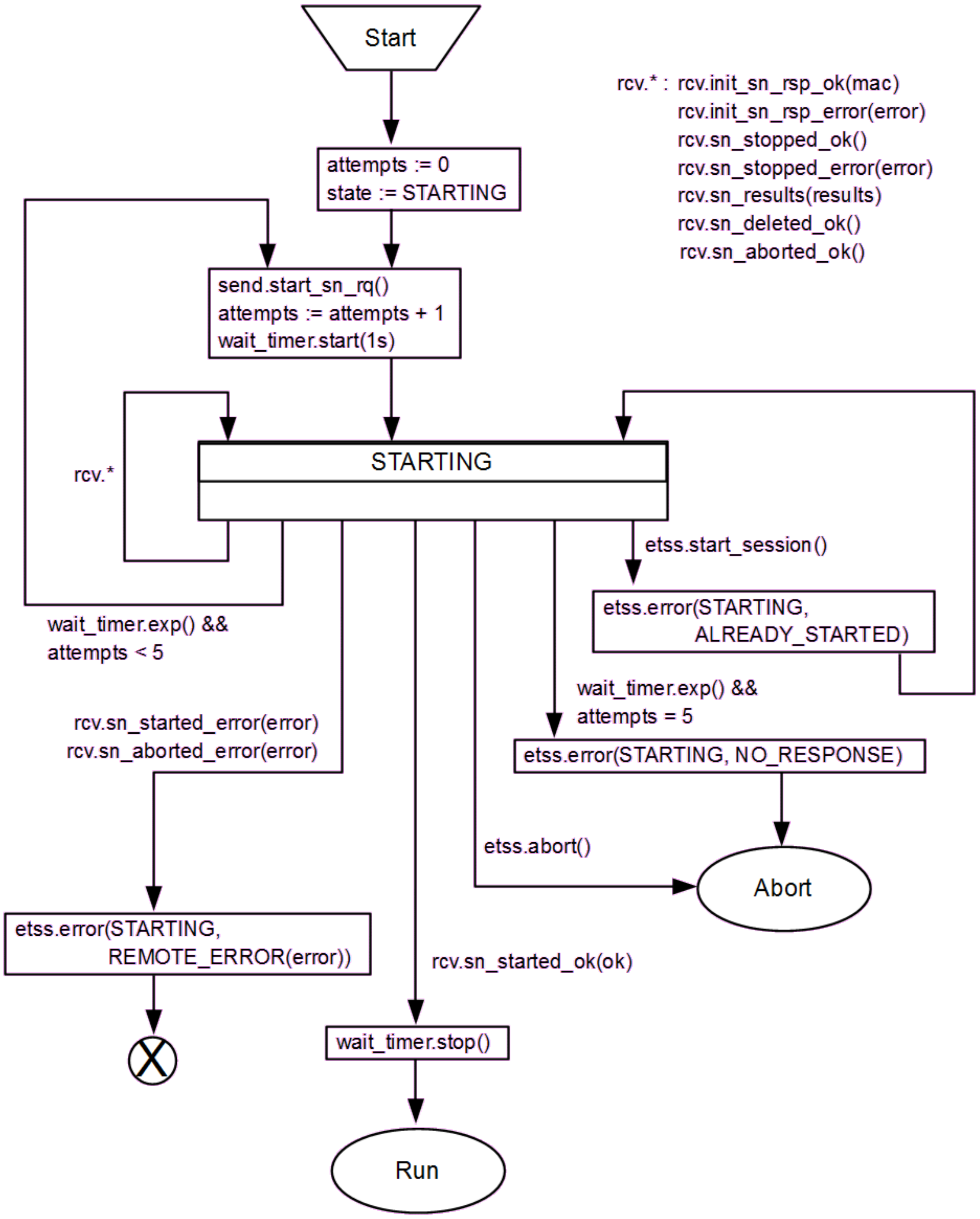

Start
attempts := 0
state := STARTING
send.start_sn_rq()
attempts := attempts + 1
wait_timer.start(1s)
rcv.* : rcv.init_sn_rsp_ok(mac)
rcv.init_sn_rsp_error(error)
rcv.sn_stopped_ok()
rcv.sn_stopped_error(error)
rcv.sn_results(results)
rcv.sn_deleted_ok()
rcv.sn_aborted_ok()
STARTING
rcv.*
etss.start_session()
etss.error(STARTING, ALREADY_STARTED)
wait_timer.exp() && attempts < 5
wait_timer.exp() && attempts = 5
rcv.sn_started_error(error)
rcv.sn_aborted_error(error)
etss.error(STARTING, NO_RESPONSE)
etss.abort()
Abort
etss.error(STARTING, REMOTE_ERROR(error))
rcv.sn_started_ok(ok)
wait_timer.stop()
Run

Figure 19 Controller End Session State Machine – Start Stage

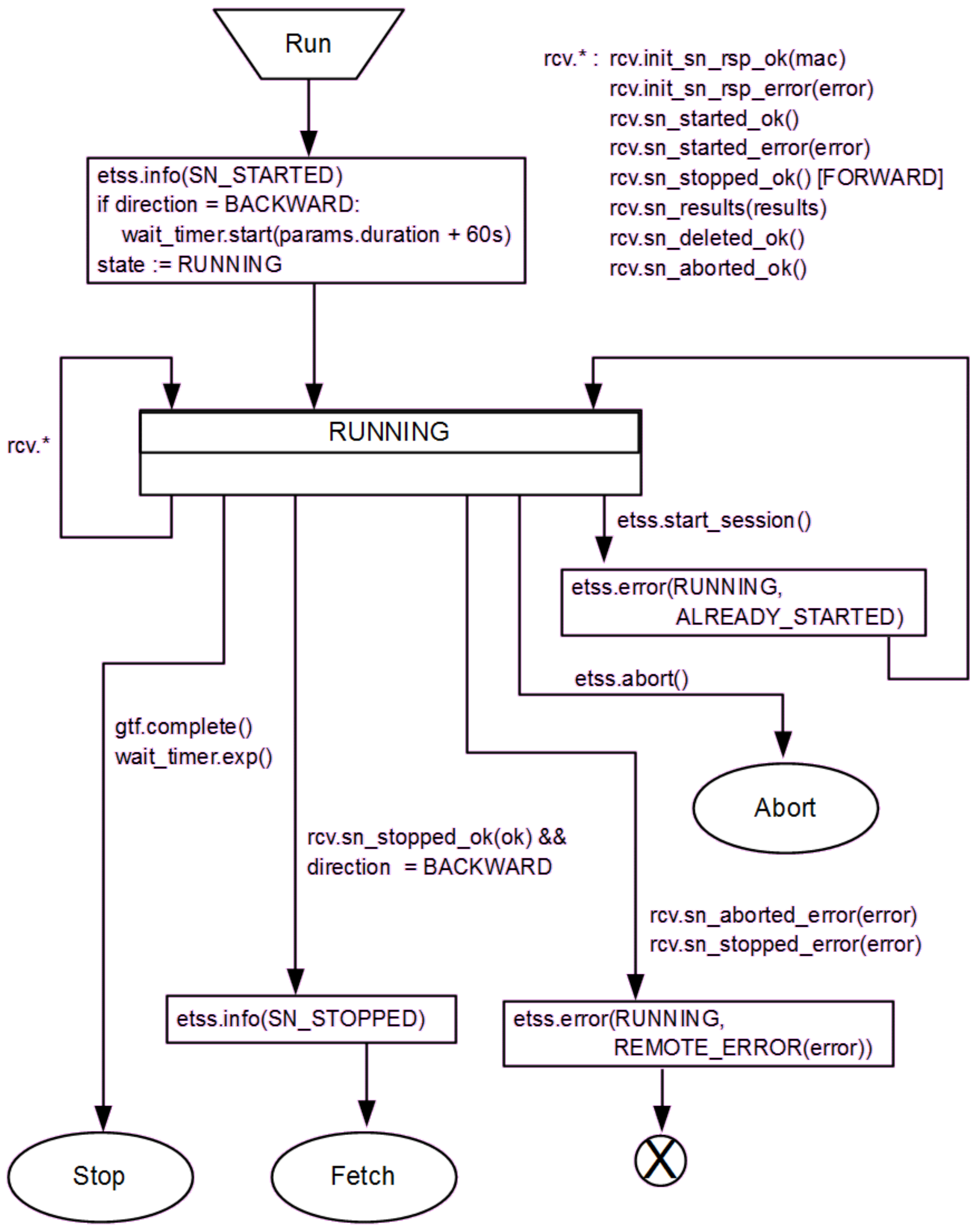

Run
etss.info(SN_STARTED)
if direction = BACKWARD:
wait_timer.start(params.duration + 60s)
state := RUNNING
rcv.* : rcv.init_sn_rsp_ok(mac)
rcv.init_sn_rsp_error(error)
rcv.sn_started_ok()
rcv.sn_started_error(error)
rcv.sn_stopped_ok() [FORWARD]
rcv.sn_results(results)
rcv.sn_deleted_ok()
rcv.sn_aborted_ok()
rcv.*
RUNNING
etss.start_session()
etss.error(RUNNING, ALREADY_STARTED)
etss.abort()
Abort
gtf.complete()
wait_timer.exp()
rcv.sn_stopped_ok(ok) && direction = BACKWARD
rcv.sn_aborted_error(error)
rcv.sn_stopped_error(error)
etss.info(SN_STOPPED)
etss.error(RUNNING, REMOTE_ERROR(error))
Stop
Fetch

Figure 20 Controller End Session State Machine – Run Stage

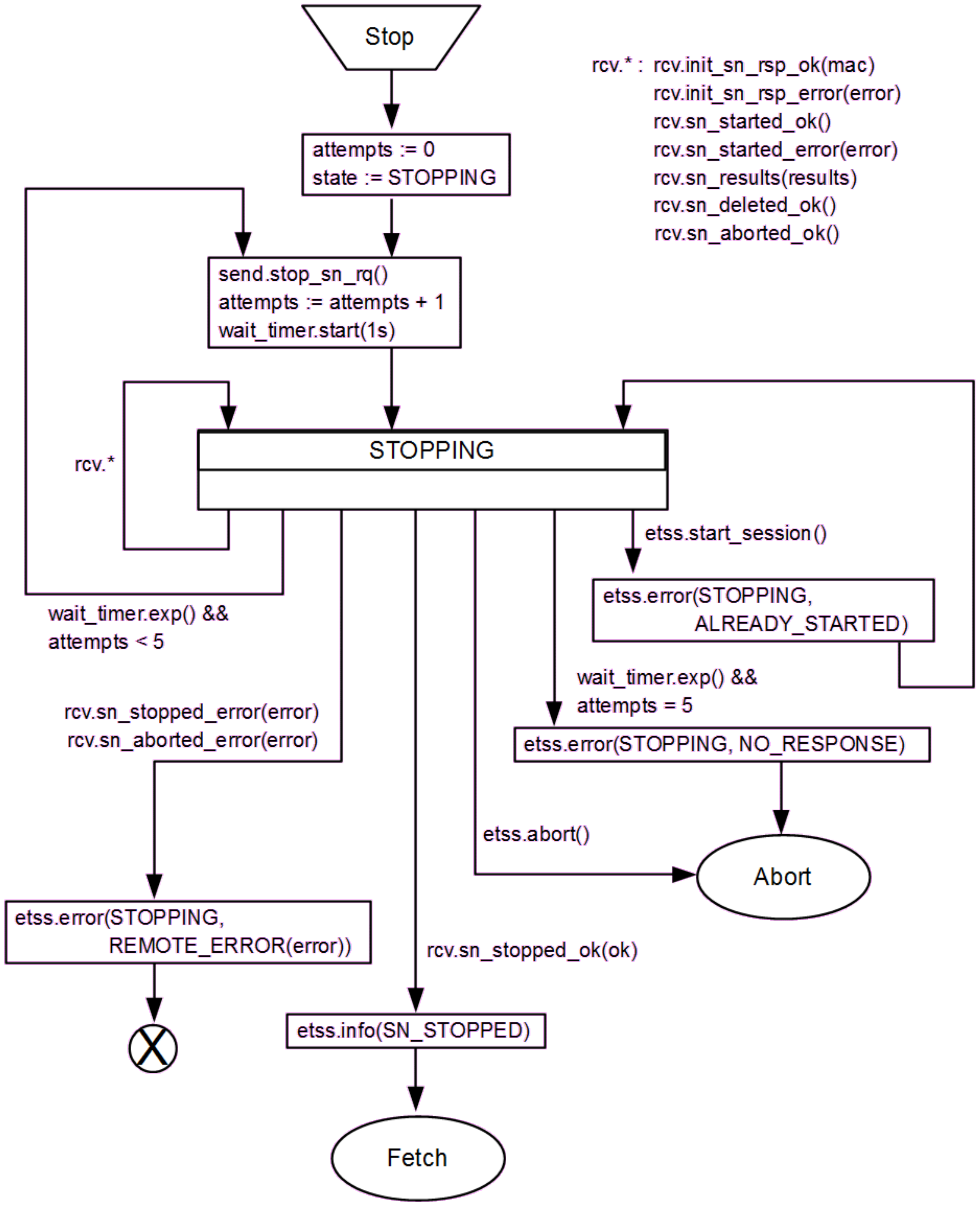

Stop
attempts := 0
state := STOPPING
rcv.* : rcv.init_sn_rsp_ok(mac)
rcv.init_sn_rsp_error(error)
rcv.sn_started_ok()
rcv.sn_started_error(error)
rcv.sn_results(results)
rcv.sn_deleted_ok()
rcv.sn_aborted_ok()
send.stop_sn_rq()
attempts := attempts + 1
wait_timer.start(1s)
rcv.*
STOPPING
etss.start_session()
etss.error(STOPPING, ALREADY_STARTED)
wait_timer.exp() && attempts < 5
wait_timer.exp() && attempts = 5
rcv.sn_stopped_error(error)
rcv.sn_aborted_error(error)
etss.error(STOPPING, NO_RESPONSE)
etss.abort()
Abort
etss.error(STOPPING, REMOTE_ERROR(error))
rcv.sn_stopped_ok(ok)
etss.info(SN_STOPPED)
Fetch

Figure 21 Controller End Session State Machine – Stop Stage

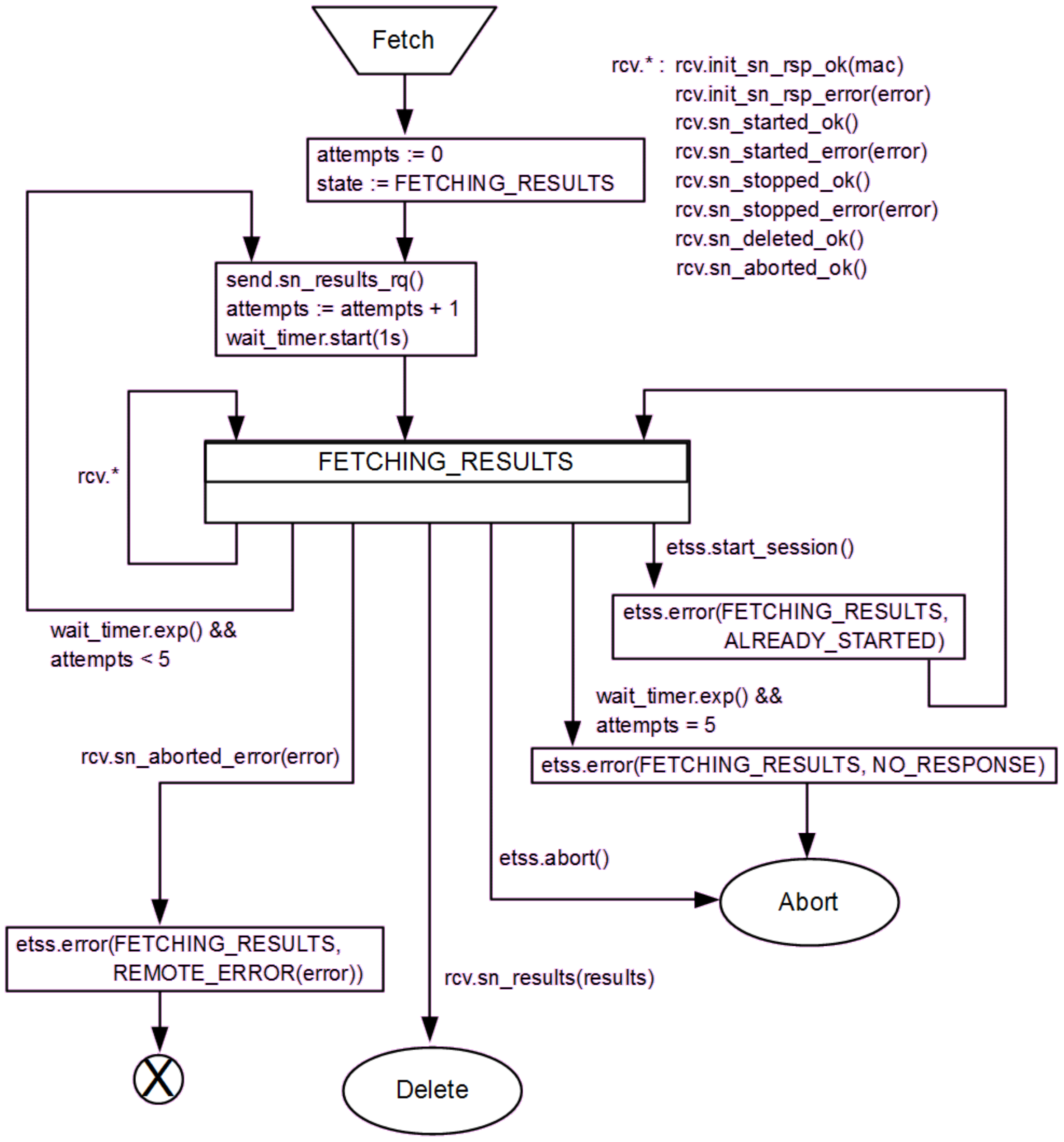

Fetch
rcv.* : rcv.init_sn_rsp_ok(mac)
rcv.init_sn_rsp_error(error)
rcv.sn_started_ok()
rcv.sn_started_error(error)
rcv.sn_stopped_ok()
rcv.sn_stopped_error(error)
rcv.sn_deleted_ok()
rcv.sn_aborted_ok()
attempts := 0
state := FETCHING_RESULTS
send.sn_results_rq()
attempts := attempts + 1
wait_timer.start(1s)
rcv.*
FETCHING_RESULTS
etss.start_session()
etss.error(FETCHING_RESULTS, ALREADY_STARTED)
wait_timer.exp() && attempts < 5
wait_timer.exp() && attempts = 5
rcv.sn_aborted_error(error)
etss.error(FETCHING_RESULTS, NO_RESPONSE)
etss.abort()
Abort
etss.error(FETCHING_RESULTS, REMOTE_ERROR(error))
rcv.sn_results(results)
Delete

Figure 22 Controller End Session State Machine – Fetch Stage

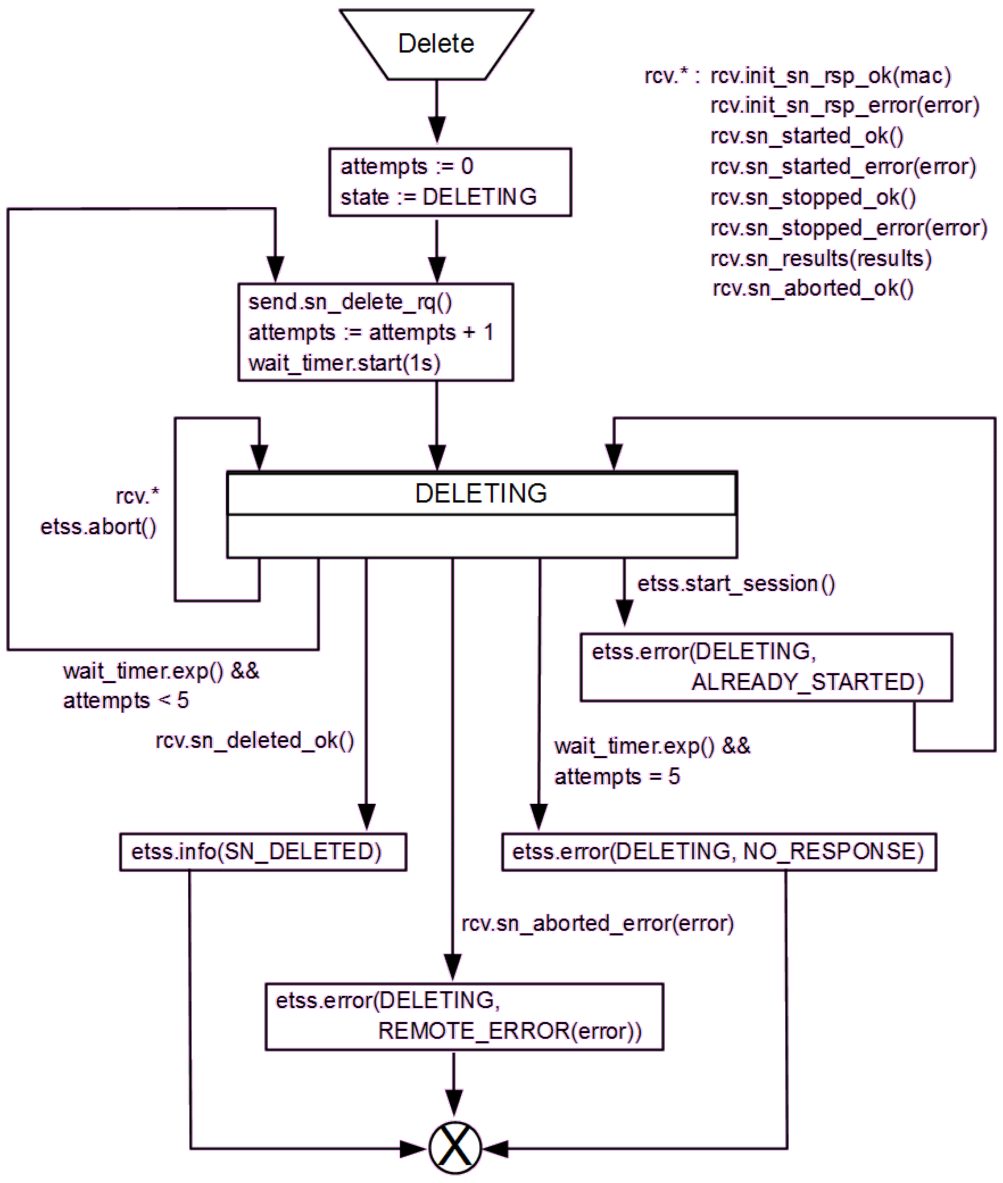

Delete
attempts := 0
state := DELETING
send.sn_delete_rq()
attempts := attempts + 1
wait_timer.start(1s)
rcv.* : rcv.init_sn_rsp_ok(mac)
rcv.init_sn_rsp_error(error)
rcv.sn_started_ok()
rcv.sn_started_error(error)
rcv.sn_stopped_ok()
rcv.sn_stopped_error(error)
rcv.sn_results(results)
rcv.sn_aborted_ok()
DELETING
rcv.*
etss.abort()
etss.start_session()
etss.error(DELETING, ALREADY_STARTED)
wait_timer.exp() && attempts < 5
rcv.sn_deleted_ok()
wait_timer.exp() && attempts = 5
etss.info(SN_DELETED)
etss.error(DELETING, NO_RESPONSE)
rcv.sn_aborted_error(error)
etss.error(DELETING, REMOTE_ERROR(error))

Figure 23 Controller End Session State Machine – Delete Stage

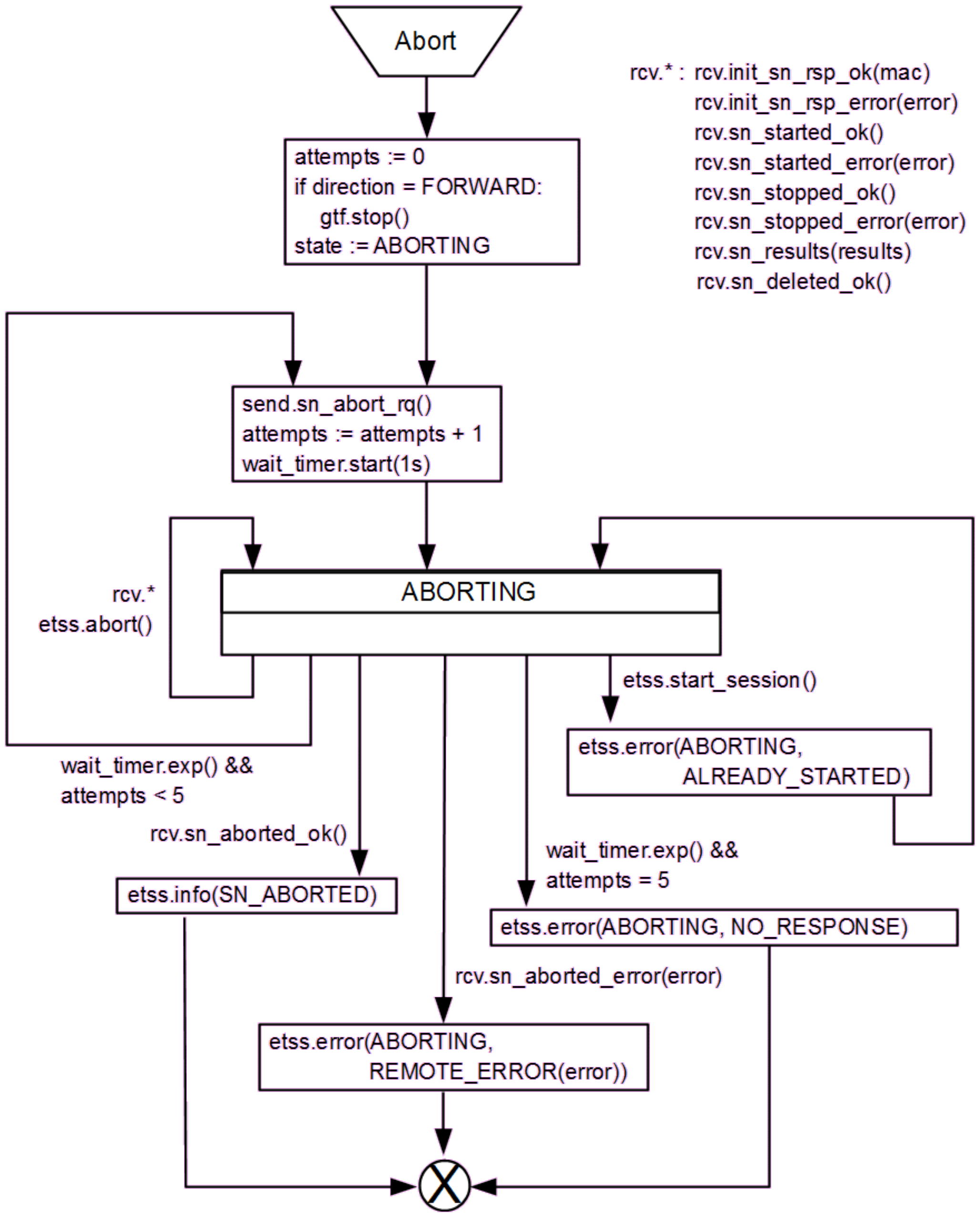

Abort
rcv.* : rcv.init_sn_rsp_ok(mac)
rcv.init_sn_rsp_error(error)
rcv.sn_started_ok()
rcv.sn_started_error(error)
rcv.sn_stopped_ok()
rcv.sn_stopped_error(error)
rcv.sn_results(results)
rcv.sn_deleted_ok()
attempts := 0
if direction = FORWARD:
gtf.stop()
state := ABORTING
send.sn_abort_rq()
attempts := attempts + 1
wait_timer.start(1s)
rcv.*
etss.abort()
ABORTING
etss.start_session()
etss.error(ABORTING, ALREADY_STARTED)
wait_timer.exp() && attempts < 5
rcv.sn_aborted_ok()
etss.info(SN_ABORTED)
wait_timer.exp() && attempts = 5
etss.error(ABORTING, NO_RESPONSE)
rcv.sn_aborted_error(error)
etss.error(ABORTING, REMOTE_ERROR(error))

Figure 24 Controller End Session State Machine – Abort Stage

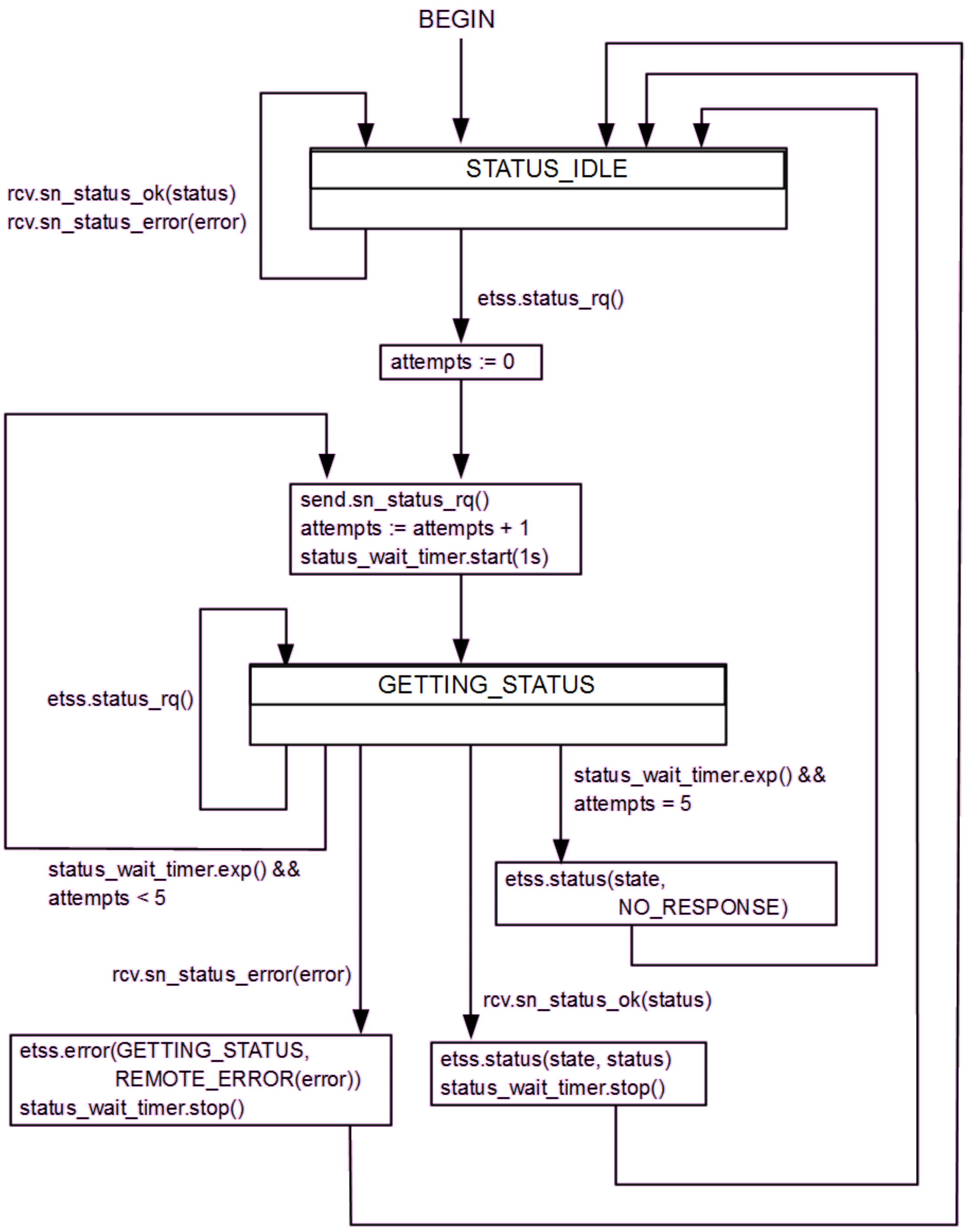

BEGIN
STATUS_IDLE
rcv.sn_status_ok(status)
rcv.sn_status_error(error)
etss.status_rq()
attempts := 0
send.sn_status_rq()
attempts := attempts + 1
status_wait_timer.start(1s)
etss.status_rq()
GETTING_STATUS
status_wait_timer.exp() && attempts = 5
status_wait_timer.exp() && attempts < 5
etss.status(state, NO_RESPONSE)
rcv.sn_status_error(error)
rcv.sn_status_ok(status)
etss.error(GETTING_STATUS, REMOTE_ERROR(error))
status_wait_timer.stop()
etss.status(state, status)
status_wait_timer.stop()

Figure 25 Controller End Session State Machine – Status stage

9.3 Responder End State Machines

A Responder End MEP contains two State Machines:

- Responder End Global State Machine (1 instance)
- Responder End Session State Machine (1 instance per active session)

For ease of description, the Responder End Session State Machine is split into several stages:

- Start
- Run
- Fetch
- Delete

At a Responder End MEP, a session (and hence an instance of the Responder End Session State Machine) is uniquely identified by the pair of (session-id, Controller End MAC) where these correspond to the Session ID and Source MAC address from a received Initiate Session Request Message. Note that the Source MAC address in the received Initiate Session Request Message is the MAC address of the Controller End MEP; this may not be the same as the MAC address of the GTF or CTF at the Controller End, which is carried in a TLV inside the message as described in section 10.2.

An overview of the Responder End State Machines is shown in Figure 26.

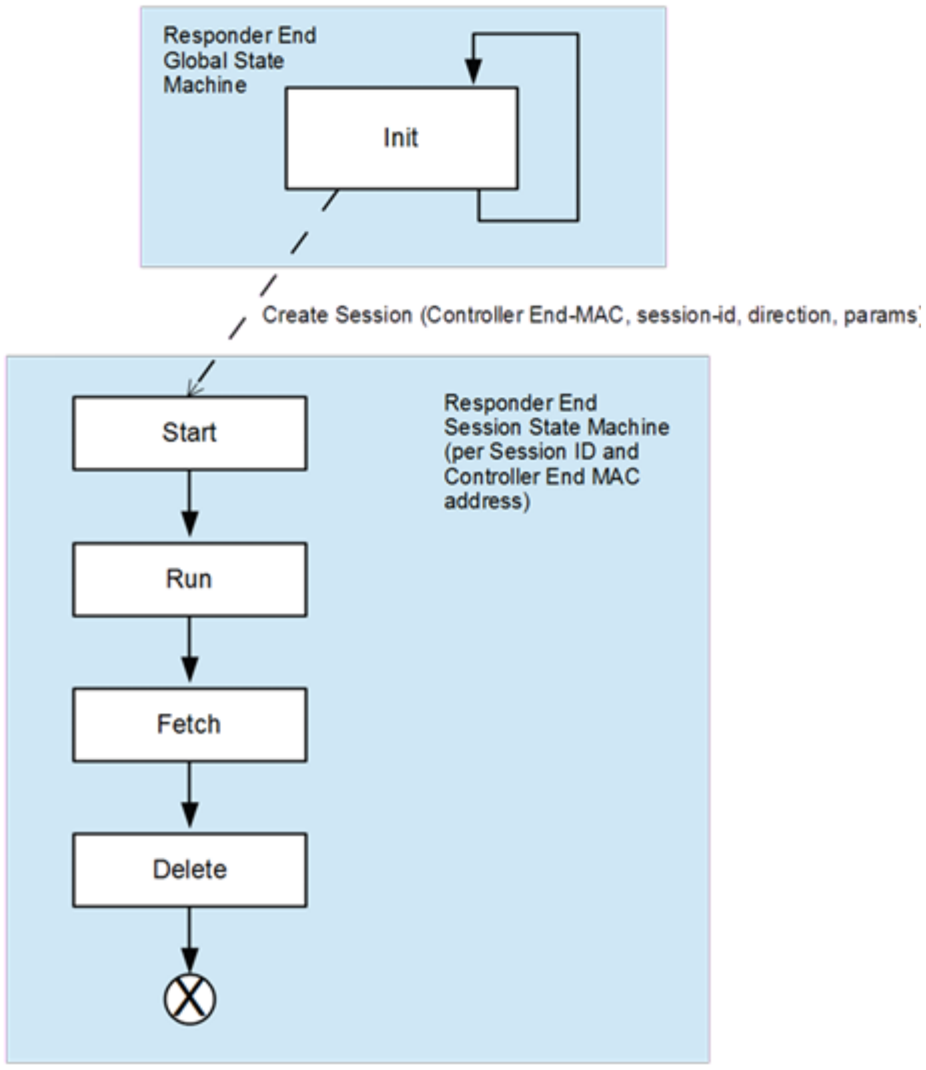


Figure 26 Responder End State Machines Overview

9.3.1 Responder End Global State Machine

There is a single instance of the Responder End Global state machine in each MEP supporting SAT Responder End functionality. This is described in the following sections.

9.3.1.1 Responder End Global State Machine Events

The Responder End Global State Machine handles the following events:

Manageability events:

- `mgmt.enable()`: A command has been received from the management layer to administratively enable the SAT Responder End.

- `mgmt.disable()`: A command has been received from the management layer to administratively disable the SAT Responder End.

Network events:

- `rcv.init_sn_rq(session_id, controller_end_mac, direction, params)`: A valid Initiate Session Request Message (that is, one conforming to the specification in section 10.2) has been received with Session ID `session_id` and source MAC address `controller_end_mac`, where there is no existing session corresponding to the (session ID, Controller End MAC) pair. The `direction` and `params`, including the SAT FS, are taken from the received message.

- `rcv.sn_status_rq(session_id, controller_end_mac)`: A valid Get Session Status Request Message (that is, one conforming to the specification in section 10.2) has been received with Session ID `session_id` and source MAC address `controller_end_mac`, where there is no existing session corresponding to the (session ID, Controller End MAC) pair.

- `rcv.unknown(session_id, controller_end_mac)`: A valid SAT PDU request message (that is, an SCM conforming to the specification in section 10.2) other than an Initiate Session Request Message or Get Session Status Request Message has been received with Session ID `session_id` and source MAC address `controller_end_mac`, where there is no existing session corresponding to the (session ID, Controller End MAC) pair.

9.3.1.2 Responder End Global State Machine Actions

The Responder End Global State Machine can take the following actions:

Network actions:

- `send.init_sn_rsp_ok(session_id, controller_end_mac)`: Send an Initiate Session Response Message to the specified `controller_end_mac`, containing the specified `session_id` and with the Response code set to 0 (NO_ERROR).

- `send.init_sn_rsp_error(session_id, controller_end_mac, error)`: Send an Initiate Session Response Message to the specified `controller_end_mac`, containing the specified `session_id` and with the Response code set to the specified `error`. Possible errors are defined in Table 7.

- `send.sn_status_error(session_id, controller_end_mac, NO_SUCH_SESSION)`: Send a Get Session Status Response Message to the specified `controller_end_mac`, containing the specified `session_id` and with the Response code set to 2 (NO_SUCH_SESSION).

- `send.sn_aborted_error(session_id, controller_end_mac, NO_SUCH_SESSION)`: Send a Abort Session Response Message to the specified `controller_end_mac`, containing the specified `session_id` and with the Response code set to 2 (NO_SUCH_SESSION).

Other actions:

- `spawn.session(session_id, controller_end_mac, direction, params)`: Spawn a Responder End Session State Machine for the given `session_id` and `controller_end_mac`, with the given `direction` and `params`.

9.3.1.3 Responder End Global State Machine Functions

The Responder End Global State Machine uses the following functions:

- `check_existing_sessions(params)`: This function checks whether there are existing sessions that could conflict with the new session. It is only used for Backward tests. If among all the MEPs that share the same GTF (and hence the same source MAC for FL-PDUs) there is an existing session for the same SAT FS, PCP and destination MAC address, this function returns a response code of 4 (TEMP_UNAVAILABLE); otherwise it returns 0 (NO_ERROR).

- `check_session(direction, params):` This function checks to see if a session with the requested `direction` and parameters can be supported at the responder end. It returns a response code of 3 (UNABLE_TO_SUPPORT) or 4 (TEMP_UNAVAILABLE) as applicable if the session cannot be supported, or 0 (NO_ERROR) otherwise. The specific checks are implementation-specific, for example to verify there are sufficient resources available to create the session.

9.3.1.4 Responder End Global State Machine

The Responder End Global state machine is shown in Figure 27.

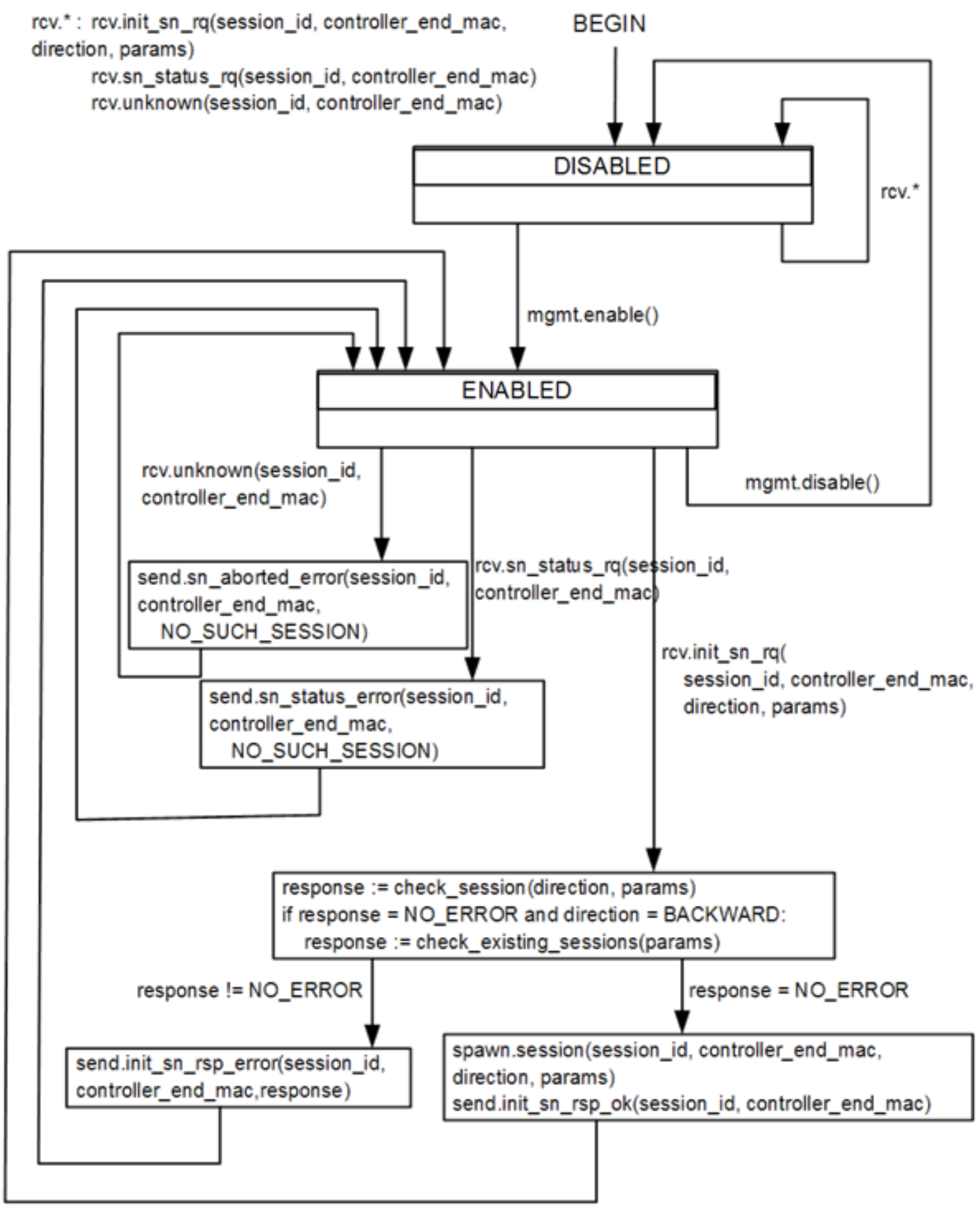

rcv.* : rcv.init_sn_rq(session_id, controller_end_mac, direction, params)
rcv.sn_status_rq(session_id, controller_end_mac)
rcv.unknown(session_id, controller_end_mac)
BEGIN
DISABLED
rcv.*
mgmt.enable()
ENABLED
mgmt.disable()
rcv.unknown(session_id, controller_end_mac)
send.sn_aborted_error(session_id, controller_end_mac, NO_SUCH_SESSION)
rcv.sn_status_rq(session_id, controller_end_mac)
send.sn_status_error(session_id, controller_end_mac, NO_SUCH_SESSION)
rcv.init_sn_rq(session_id, controller_end_mac, direction, params)
response := check_session(direction, params)
if response = NO_ERROR and direction = BACKWARD:
response := check_existing_sessions(params)
response != NO_ERROR
response = NO_ERROR
send.init_sn_rsp_error(session_id, controller_end_mac,response)
spawn.session(session_id, controller_end_mac, direction, params)
send.init_sn_rsp_ok(session_id, controller_end_mac)

Figure 27 Responder End Global State Machine

9.3.2 Responder End Session State Machines

There is an instance of the Responder End Session State Machine in the Responder End MEP for each unique (Session ID, Controller End MAC) pair that has been created by the Responder End Global State Machine in that MEP. An instance of the Session State Machine is associated with exactly one (session ID, Controller End MAC) pair. The instance continues running until it terminates according to the specification below. When it terminates, all resources are released and all collected results are discarded.

The instance also has an associated direction (FORWARD or BACKWARD) and session parameters, as specified when the instance is spawned. The CTF (for a Forward session) or GTF (for a Backward session) is created at the same time as the state machine is spawned, and is started as shown in the state machine diagrams. Similarly, the GTF or CTF is deleted and any resources released when the state machine terminates.

All events and actions, except for `mgmt.disable()`, are specific to the MEP, session ID and Controller End MAC for which the state machine instance is running. Note that the MEP, session ID and Controller End MAC are not shown explicitly as parameters to the State Machine events and actions, but may be presumed implicitly. In addition, the state machine instance uses the session-id, direction and params that were specified when it was spawned.

9.3.2.1 Responder End Session State Machine Events

The Responder End Session State Machine handles the following events:

Manageability events:

- `mgmt.disable()`: A command has been received from the management layer to globally administratively disable the SAT Responder.

Network events:

- `rcv.init_sn_rq(direction, params)`: A valid Initiate Session Request Message (that is, one conforming to the specification in section 10.2) has been received for the session, containing the specified `direction` and `params`.
- `rcv.start_sn_rq()`: A valid Start Session Request Message (that is, one conforming to the specification in section 10.2) has been received for the session.
- `rcv.stop_sn_rq()`: A valid Stop Session Request Message (that is, one conforming to the specification in section 10.2) has been received for the session.
- `rcv.sn_status_rq()`: A valid Get Session Status Request Message (that is, one conforming to the specification in section 10.2) has been received for the session.

- `rcv.sn_results_rq()`: A valid Get Session Results Request Message (that is, one conforming to the specification in section 10.2) has been received for the session.

- `rcv.sn_delete_rq()`: A valid Delete Session Request Message (that is, one conforming to the specification in section 10.2) has been received for the session.

- `rcv.sn_abort_rq()`: A valid Abort Session Request Message (that is, one conforming to the specification in section 10.2) has been received for the session.

Other events:

- `wait_timer.exp()`: The Wait Timer has expired

- `gtf.complete()`: The GTF has finished sending all the frames for the session.

9.3.2.2 Responder End Session State Machine Actions

The Responder End Session State Machine can take the following actions:

Manageability actions:

- `mgmt.error(error)`: The manageability system is notified of the specified `error` having occurred. Possible errors include:

  - NO_START: No Start Session Request Message was received within the wait time (for Backward tests only).

  - NO_STOP: No Stop Session Request Message was received within the wait time (for Forward tests only).

  - NO_FETCH: No Get Session Results Request Message was received within the wait time.

  - NO_DELETE: No Delete Session Request Message was received within the wait time.

  - ABORT: An Abort Session Request Message was received from the Controller End.

  - UNEXP_RQ: An unexpected request message was received.

- `mgmt.info(event)`: The manageability system is optionally notified of the specified `event` having occurred. Possible events include:

  - SN_CREATED

  - SN_DELETED

Network actions:

- `send.init_sn_rsp_ok()`: Send an Initiate Session Response Message for the session, with the Response code set to 0 (NO_ERROR).
- `send.init_sn_rsp_error(error)`: Send an Initiate Session Response Message for the session, with the Response code set to the specified `error`. Possible errors are defined in Table 7.
- `send.sn_started_ok()`: Send a Start Session Response Message for the session, with the Response code set to 0 (NO_ERROR).
- `send.sn_started_error(error)`: Send a Start Session Response Message for the session, with the Response code set to the specified `error`. Possible errors are defined in Table 7.
- `send.sn_stopped_ok()`: Send a Stop Session Response Message for the session, with the Response code set to 0 (NO_ERROR).
- `send.sn_stopped_error(error)`: Send a Stop Session Response Message for the session, with the Response code set to the specified `error`. Possible errors are defined in Table 7.
- `send.sn_status(status)`: Send a Get Session Status Response Message for the session, containing the given `status,` and the response code set to 0 (NO_ERROR). Possible status values are defined in Table 12.
- `send.sn_results(results)`: Send a Get Session Results Response Message for the session, containing the session `results,` and the response code set to 0 (NO_ERROR).
- `send.sn_deleted_ok()`: Send a Delete Session Response Message for the session, with the Response code set to 0 (NO_ERROR).
- `send.sn_aborted_ok()`: Send an Abort Session Response Message for the session, with the Response code set to 0 (NO_ERROR).
- `send.sn_aborted_error(error)`: Send an Abort Session Response Message for the session, with the Response code set to the specified `error`. Possible errors are defined in Table 7.

Other actions:

- `wait_timer.start(duration)`: Start the wait timer for the specified `duration`
- `wait_timer.stop()`: Stop the wait timer.

- `ctf.start(params)`: Start the CTF, for the given SAT FS, PCP, source MAC and (for a broadcast or multicast test) destination MAC in the given parameters. For a unicast test, the destination MAC is the CTF's own MAC address.

Note that the source MAC is the MAC address of the GTF at the Controller End, extracted from the session parameters. This may not be the same as the Controller End MEP's MAC address, which is used to identify the session at the Responder End.

- `gtf.start(params):` Start the GTF, for the SAT FS, PCP and destination MAC address in the given parameters. The source MAC address is the GTF's MAC address.

9.3.2.3 Responder End Session State Machine

The Responder End Session State Machine is shown in Figure 28, Figure 29, Figure 30, and Figure 31.

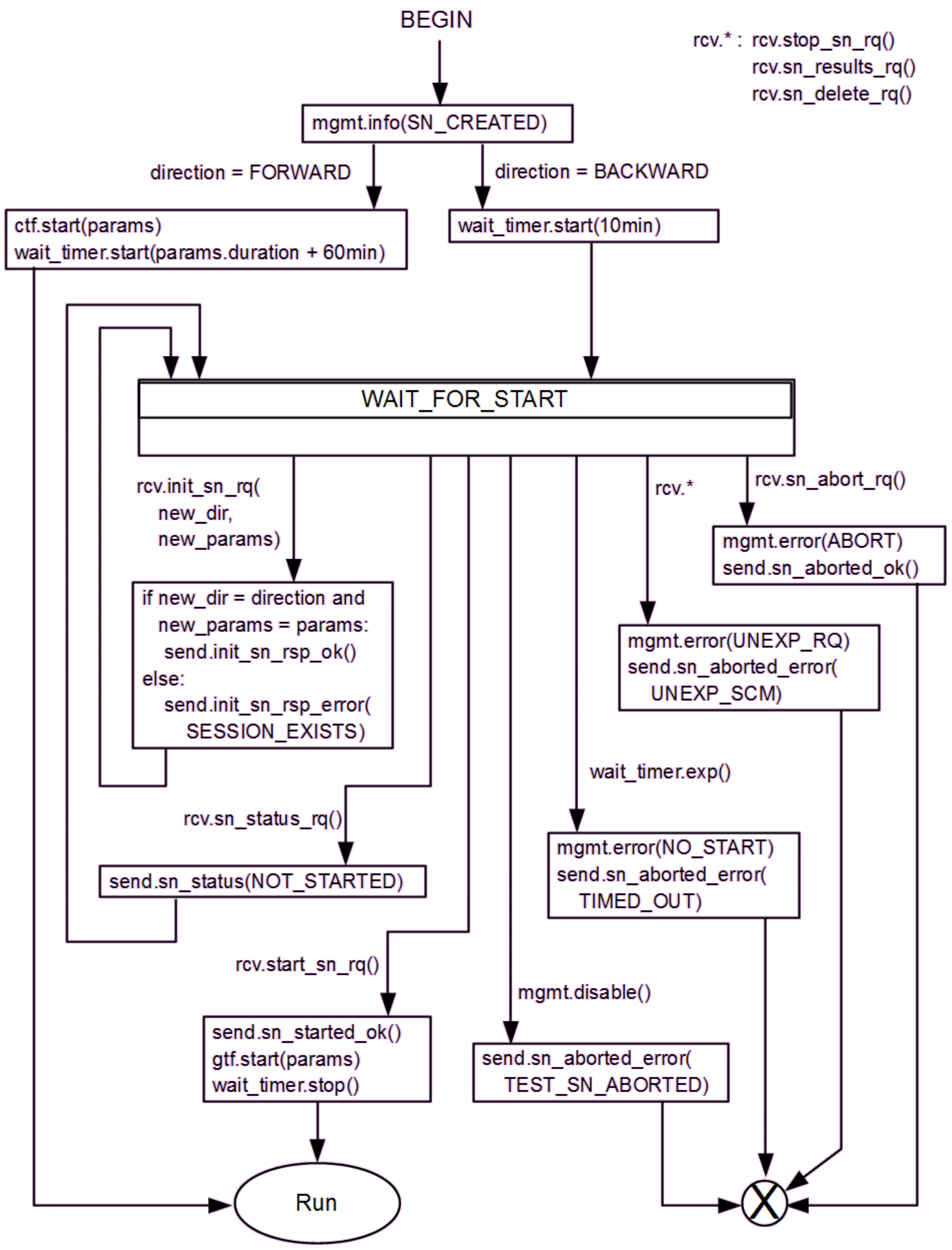


Figure 28 Responder End Session State Machine – Start stage

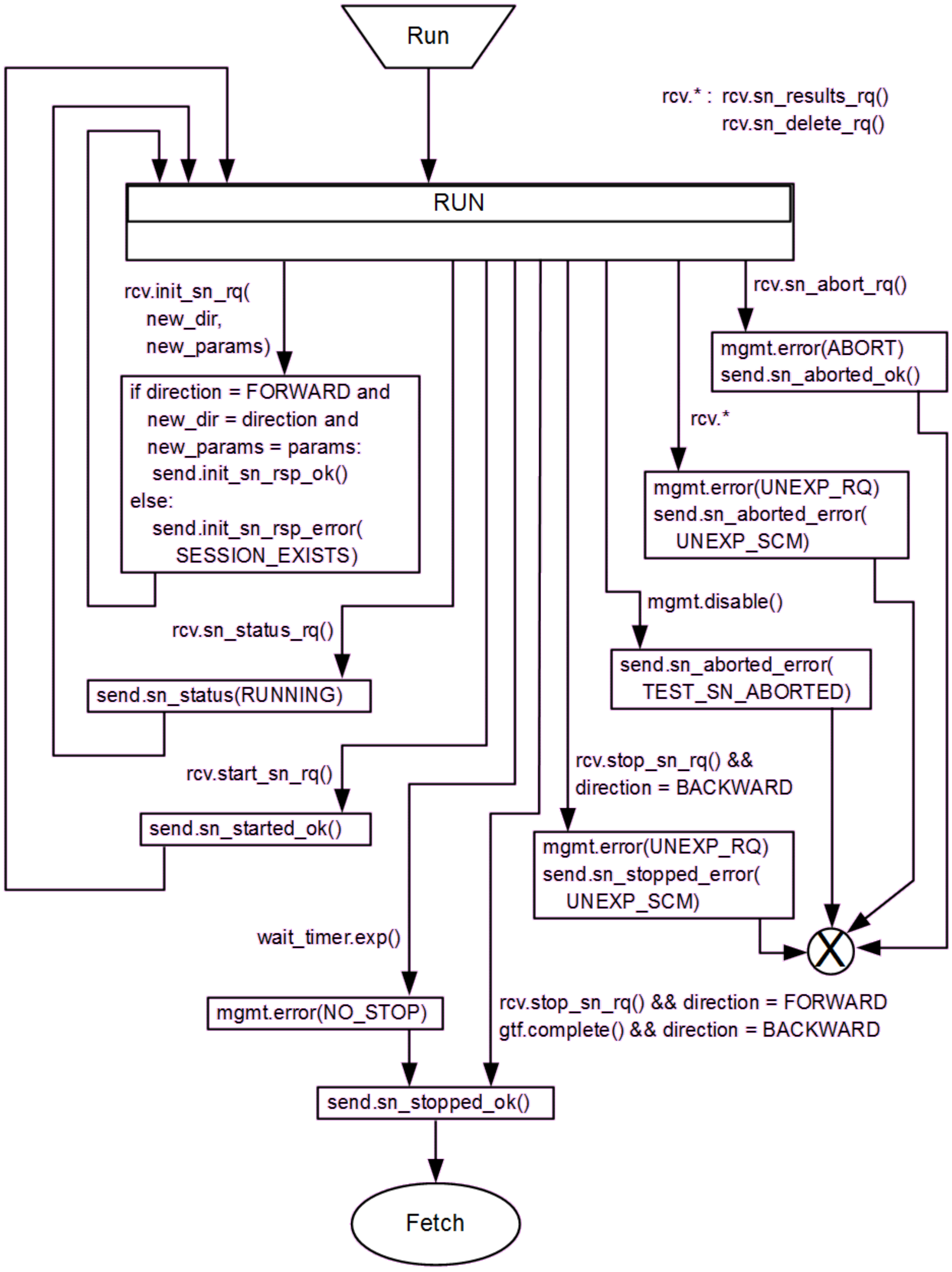


Figure 29 Responder End Session State Machine – Run stage

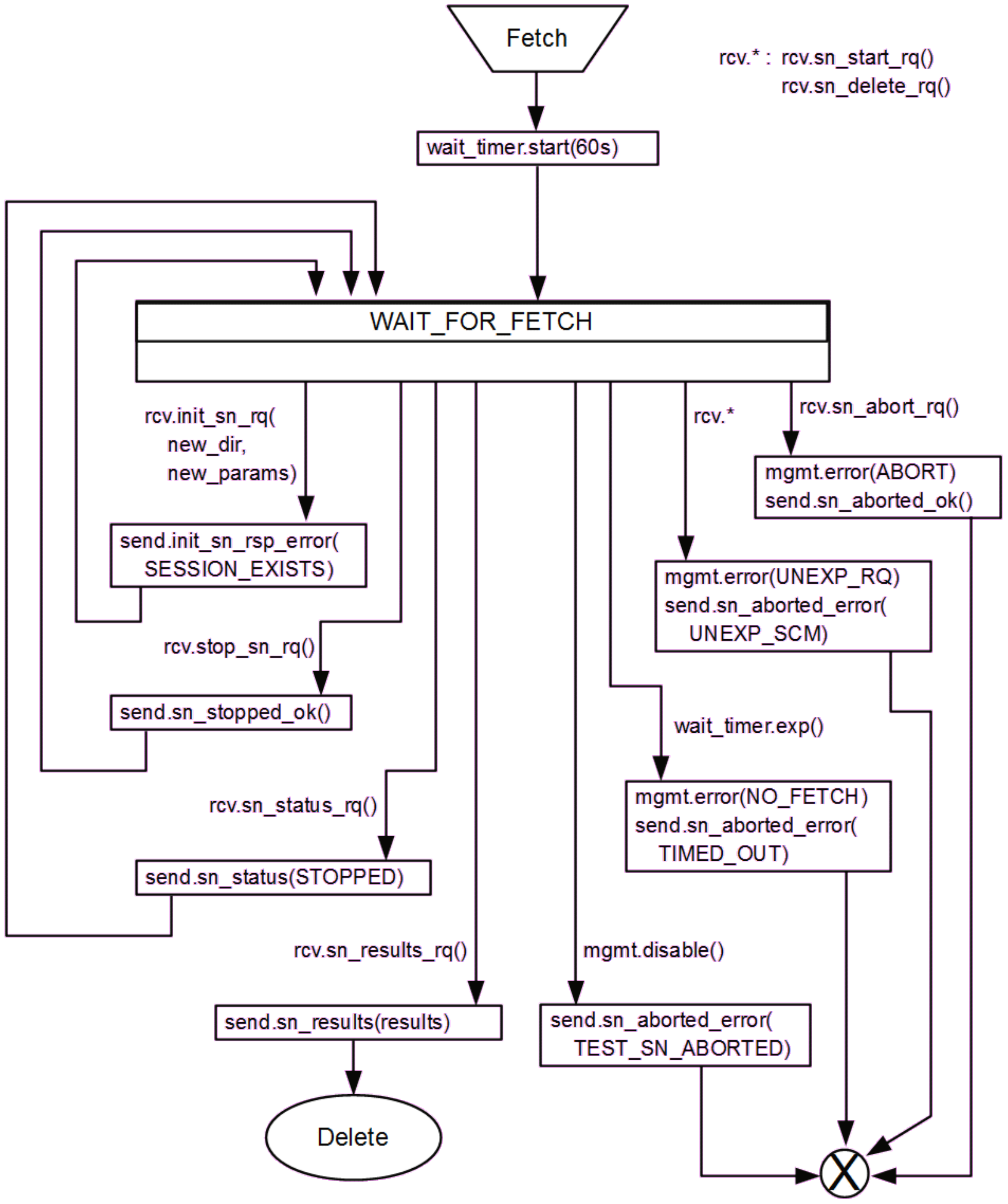

Fetch
rcv.* : rcv.sn_start_rq()
rcv.sn_delete_rq()
wait_timer.start(60s)
WAIT_FOR_FETCH
rcv.init_sn_rq(new_dir, new_params)
send.init_sn_rsp_error(SESSION_EXISTS)
rcv.stop_sn_rq()
send.sn_stopped_ok()
rcv.sn_status_rq()
send.sn_status(STOPPED)
rcv.sn_results_rq()
send.sn_results(results)
Delete
mgmt.disable()
send.sn_aborted_error(TEST_SN_ABORTED)
rcv.*
mgmt.error(UNEXP_RQ) send.sn_aborted_error(UNEXP_SCM)
wait_timer.exp()
mgmt.error(NO_FETCH) send.sn_aborted_error(TIMED_OUT)
rcv.sn_abort_rq()
mgmt.error(ABORT) send.sn_aborted_ok()

Figure 30 Responder End Session State Machine – Fetch stage

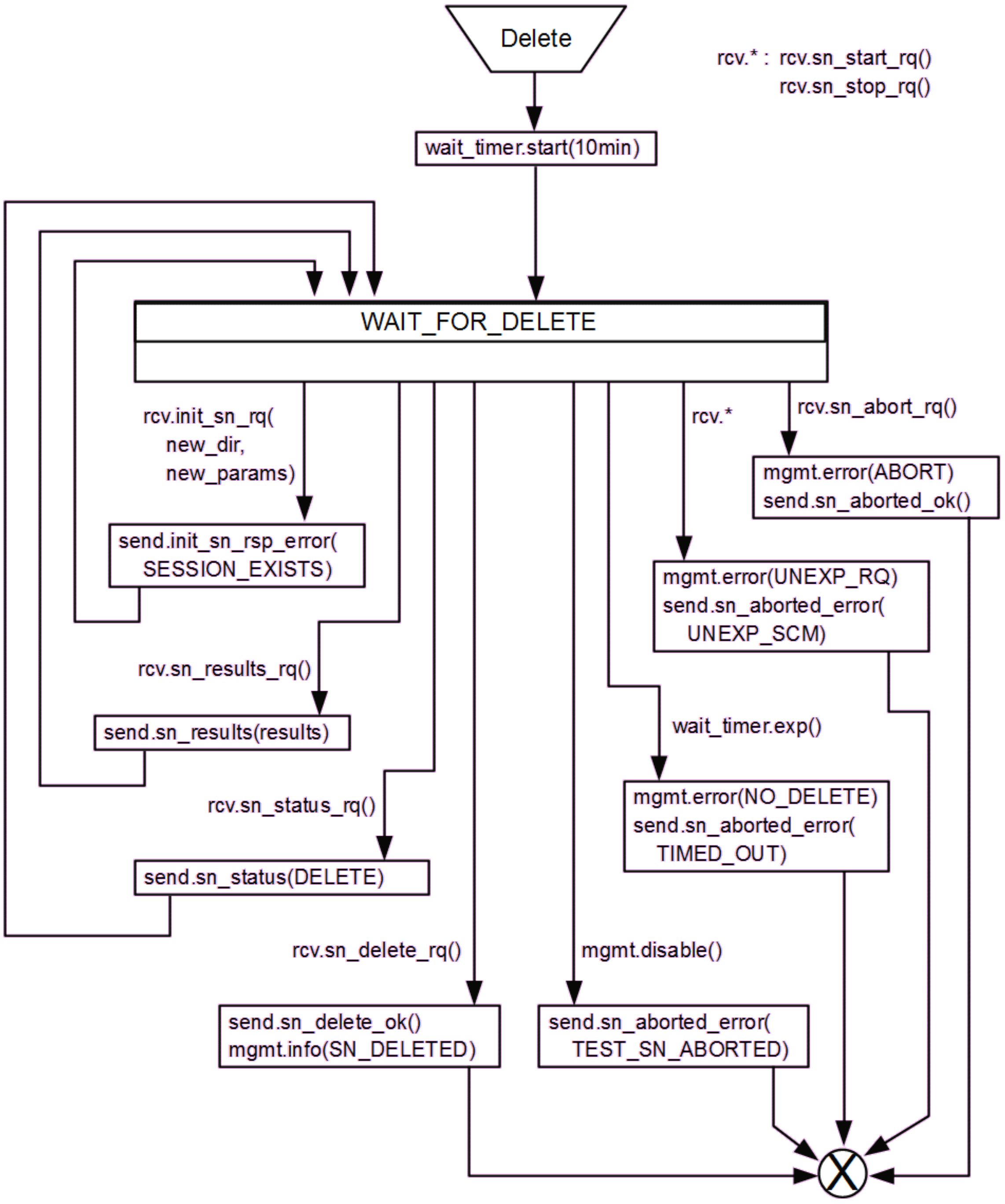

Delete
rcv.* : rcv.sn_start_rq()
rcv.sn_stop_rq()
wait_timer.start(10min)
WAIT_FOR_DELETE
rcv.init_sn_rq(new_dir, new_params)
send.init_sn_rsp_error(SESSION_EXISTS)
rcv.sn_results_rq()
send.sn_results(results)
rcv.sn_status_rq()
send.sn_status(DELETE)
rcv.sn_delete_rq()
send.sn_delete_ok() mgmt.info(SN_DELETED)
mgmt.disable()
send.sn_aborted_error(TEST_SN_ABORTED)
wait_timer.exp()
mgmt.error(NO_DELETE) send.sn_aborted_error(TIMED_OUT)
rcv.*
mgmt.error(UNEXP_RQ) send.sn_aborted_error(UNEXP_SCM)
rcv.sn_abort_rq()
mgmt.error(ABORT) send.sn_aborted_ok()

Figure 31 Responder End Session State Machine – Delete stage

10. SAT Control Protocol PDU

The SAT Control Protocol uses SAT Control Protocol Message (SCM) and SAT Control Protocol Response (SCR) PDUs as defined in this document. These are ITU-T G.8013/Y.1731 [3] PDUs and use the G.8013 EtherType (0x8902) and encapsulation. The fields of each message are further explained below. When consecutive octets are used to represent a binary number, the lower octet number has the most significant value. The bits in an octet are numbered from 1 to 8, where bit 1 is the least significant bit (LSB) and bit 8 is the most significant bit (MSB). SCMs and SCRs are transmitted marked with the CoS ID supported by the service under test that offers the lowest frame loss objective.

[R40] A received PDU **MUST** be processed and validated as described in G.8013/Y.1731 [3] clauses 11.2 and 11.3.

[R41] If an SCM is received by the Responder End that does not comply with the requirements within this Section, the Responder End **MUST** reply with an SCR containing an Abort Session Response message with a Response Code of 1 (MALFORMED_RQ).

10.1 SAT Frame Set

As frames flow between two given EIs, they may experience VLAN tagging operations at a number of places. Tagging operations include pushing and popping tags, and changing the VLAN-ID in a tag (VLAN translation). Such operations may occur within an EI (including between a GTF or CTF and the corresponding Controller or Responder MEP) or between EIs within a CEN.

Therefore, a given flow – i.e., a set of frames – cannot be identified simply with a particular number of VLAN tags or set of VLAN-IDs; it is also necessary to specify the exact point inside an EI at which the frames in the set have that number of VLAN tags with those VLAN IDs.

In the context of Service Activation Testing, it is essential that the set of frames generated by the GTF is the same as the set of frames counted by the CTF. This set of frames is part of the SAT Frame Set (SAT FS) for a given test session. The information about the SAT FS must be conveyed from the Controller MEP to the Responder MEP, so that the Responder MEP can correctly configure its local GTF (for a Backward session) or CTF (for a Forward session).

However, as described above, the frames in a SAT FS may have different VLAN tags at the Controller End and the Responder End. It is not therefore possible to carry the information about the VLAN tags within the body of the Control Protocol's Initiate Session PDU, as is done for all other session parameters; as above, a frame set is only identified by a given set of VLAN tag information at a specific point within an EI. Instead, the SAT FS for the session is derived from the tags carried in the Ethernet header of the Control Protocol frame itself. In this way, any VLAN tagging operations on test frames in the SAT FS that occur between the Controller End and the Responder End will also affect the Control Protocol frames.

A SAT Frame Set may therefore be defined as a set of frames generated by a GTF or counted by a CTF within a particular EI, identified by containing the same CE-VLAN ID and/or S-VLAN ID, or by being untagged; that is, that fall into one of the following four (a, b, c, or d) mutually exclusive cases:

a) untagged; or
b) containing a C-tag with a particular CE-VLAN ID at a UNI, and not containing an S-tag; or
c) containing an S-tag with a particular S-VLAN ID, and not containing a C-tag, at an ENNI that:
    a. does not have a VUNI associated with that S-VLAN ID; or
    b. has a VUNI associated with that S-VLAN ID, if the frame is generated towards or received over the ENNI; or
    c. has a VUNI associated with that S-VLAN ID, if the VUNI has an OVC End Point matching un-C-tagged frames;

    or
d) containing an S-tag with a particular S-VLAN ID at an ENNI that has a VUNI associated with that S-VLAN ID, and containing a C-tag with a particular CE-VLAN ID.

Received SAT Control Frames may be consumed by a MEP, before they reach the CTF; however, they are considered to belong to a SAT FS according to the tags they would have when passing through the CTF, if the consuming MEP did not exist. Similarly, SAT Control Frames sent by a MEP are considered to belong to a SAT FS according to the tags they would need to have been generated with at the GTF, such that they would be transmitted with the same tags if the MEP did not exist. In other words, even within a single EI, it is possible that VLAN tagging operations occur between the GTF/CTF and the Controller or Responder MEP (see Figures 1 to 6). To account for this, the SAT FS for Control Protocol frames is not determined by the tags they have at the point where they are generated or received at a MEP, but by the tags they would have had at the point where they pass through the GTF or CTF, if the MEPs were not there – or in other words, if the MEG were infinitely wide.

For example, consider an ENNI-N, with an Up MEP in the operator MEG. Suppose that 802.1Q Provider Bridging is used within the CEN, with S-VLAN ID 100 used to carry a particular EVC within the CEN; but that S-VLAN ID 207 is used for that EVC across the ENNI. In this case, the EEIF within the ENNI-N translates one S-VLAN ID to another (see [6]).

In this case, the MEP sends and receives Control Protocol Frames towards the Peer MEP with S-VLAN ID 100, as the MEP is located within the ENNI-N before (i.e., on the CEN side of) the EEIF. However, test frames passing through the MEP with S-VLAN 100 will have been generated by the GTF, or will reach the CTF, with S-VLAN 207, as the VLAN is translated at the EEIF, and the GTF and CTF are located after (i.e., on the ENNI side of) the EEIF. This is shown in the figure below.

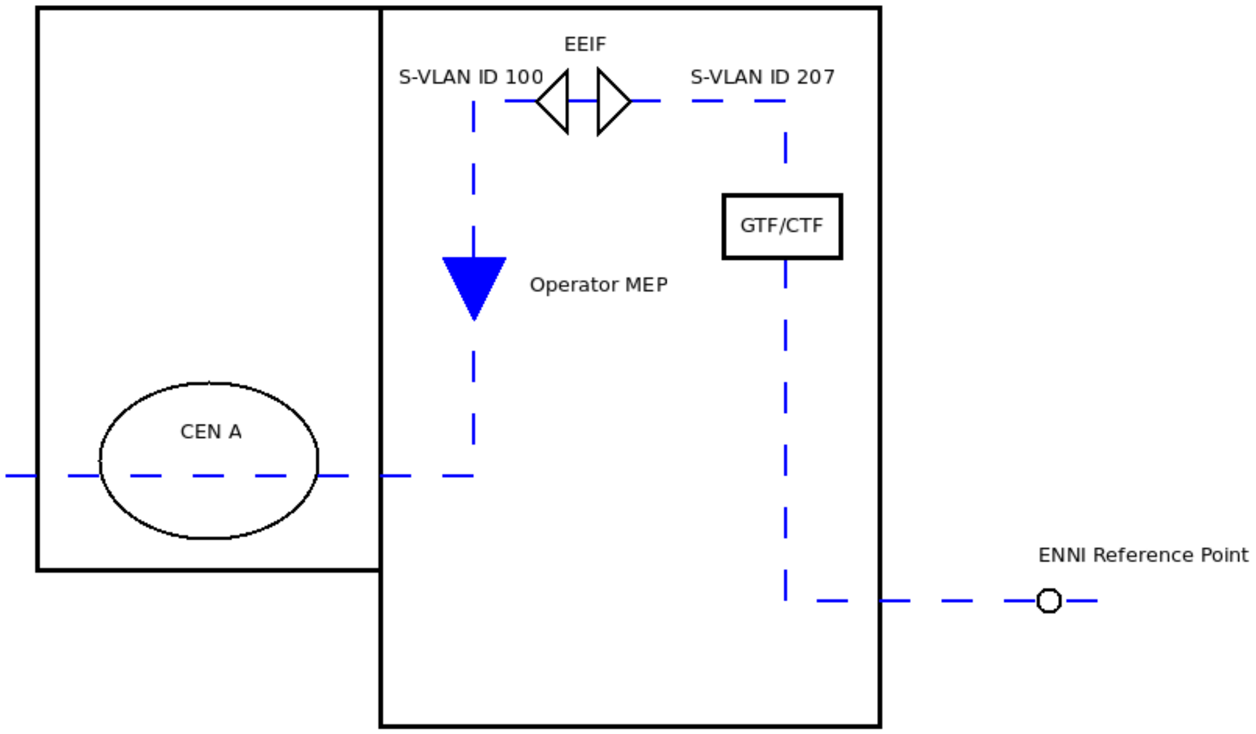


Figure 32 SAT FS VLAN ID Translation

The SAT FS for a test session in this example is identified by having S-VLAN ID 207. The Control Protocol frames with S-VLAN ID 100 that are generated or received by the MEP also belong to this frame set, because they would have had S-VLAN ID 207 when they reached the CTF, had they not been consumed by the MEP.

[R42] For a given test session, a Controller or Responder MEP **MUST** send Control Protocol frames that belong to the SAT FS for the session.

As described above, this requirement does not mean that the Control Protocol frames necessarily have exactly the same tags as those used to identify the SAT FS at the GTF or CTF.

Note: A MEP cannot be used as the Controller End MEP for a Test Session if, given the SAT FS for the Test Session, [R42] would necessitate the MEP transmitting frames with a VID that is not in the list of VIDs associated with the MEP's MEG (or would necessitate the MEP transmitting tagged frames, if the MEP's MEG is not associated with any VIDs).

10.2 SAT Control Protocol Message

The SAT Control Protocol Message (SCM) is used by the Controller End to send requests to the Responder End. The SCM is defined below.

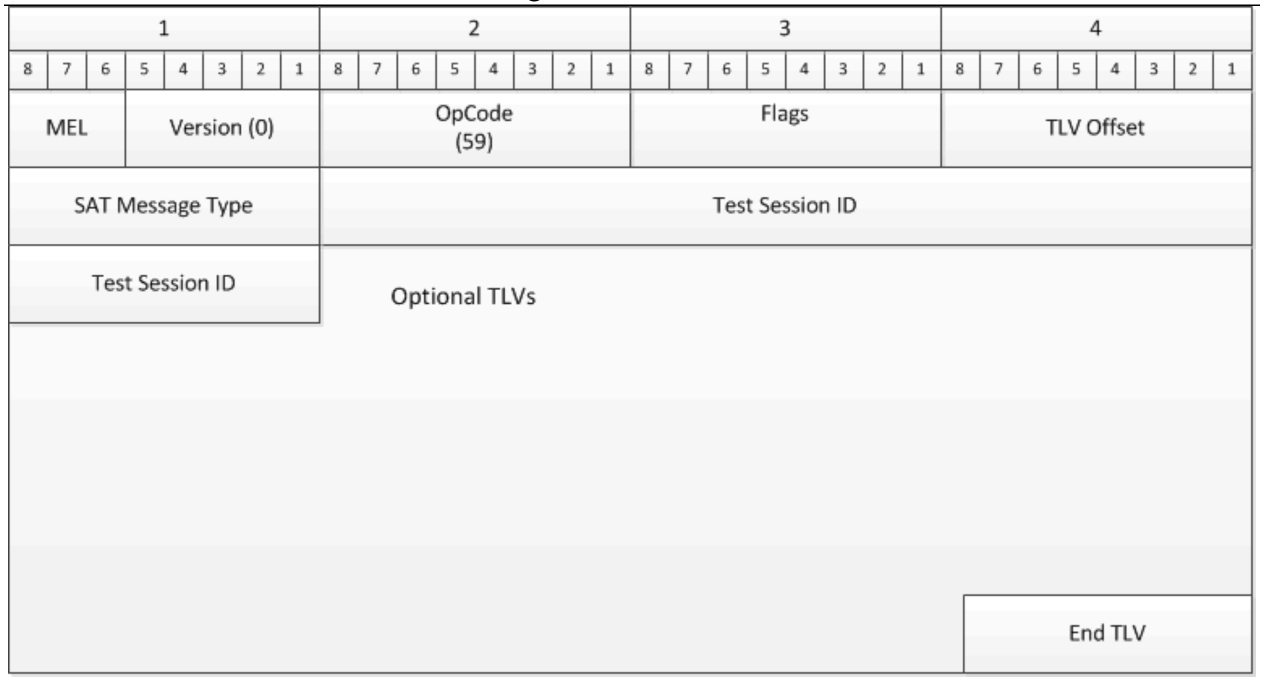


Figure 33 Generic SAT Control Protocol Message Format

[R43] Fixed fields in SCMs **MUST** be located as shown in Figure 33.

10.2.1 MEL

The 3-bit MEL field contains the MEG Level of the MEP to which the message is directed.

[R44] The MEL **MUST** be set to the MEG Level used for SAT.

10.2.2 Version

The 5-bit Version field contains the version of the SCM.

[R45] The Version **MUST** be set to 0 in transmitted SCM frames.

[R46] A MEP **MUST** process and not discard the PDU according to the Version in the PDU or the Version supported by the MEP, whichever is lower, as described in G.8013/Y.1731 [3] clauses 11.2 and 11.3.

10.2.3 OpCode

The value used in the OpCode field is within the range allocated in ITU-T G.8013/Y.1731 [3] for use by MEF.

[R47] The OpCode for the SCM **MUST** be set to 59.

10.2.4 Flags

The Flags field is used in the Initiate Session Request Message to indicate whether a Forward or Backward session is required.

[R48] In the Initiate Session Request Message bit 8 of the Flags field **MUST** be set to 0 when requesting a Forward session from the Responder End.

[R49] In the Initiate Session Request Message bit 8 of the Flags field **MUST** be set to 1 when requesting a Backward session from the Responder End

[R50] In the Initiate Session Request Message all other bits in the Flags field are reserved for future use and **MUST** be set to 0 on transmit and ignored on receipt.

[R51] In all other message types sent by the Controller End the Flags field **MUST** be set to 0 and ignored on receipt.

10.2.5 TLV Offset

The TLV Offset indicates the number of octets from this field at which the TLV fields, if any, begin.

[R52] The TLV Offset **MUST** be set to indicate the start of the first TLV.

[R53] The transmitted TLV Offset **MUST** be 5.

[R54] A received SCM **MUST** have a TLV Offset of 5 or greater.

[R55] The TLV Offset **MUST** either indicate a location for the first TLV that is within the SCM PDU, or must indicate a location that corresponds to the first octet of the FCS if this immediately follows the SCM PDU in the frame.

Note: In the latter case, where the TLV Offset indicates a location that corresponds to the first octet of the FCS, then there are no TLVs (including no END TLV).

10.2.6 Message Type

The values for the Message Type are shown in Table 4.

Message Type	Message Name	Description
1	Initiate Session Request Message	Requests a test session be initiated
2	Start Session Request Message	Requests a test session be started
3	Stop Session Request Message	Requests a test session be stopped
4	Abort Session Request Message	Requests an in-progress test session be aborted
5	Get Session Status Request Message	Requests status on a test session
6	Fetch Session Results Request Message	Requests results from a test session
7	Delete Session Request Message	Identifies that a test session is complete and the test results at the Responder End can be deleted

Table 4 SAT Control Protocol SCM Message Types

[R56] The Message Type in the SCM **MUST** be set as defined in Table 4.

[R57] All other values are reserved and SCMs containing them **MUST** be ignored.

10.2.7 Test Session ID

The Test Session ID field contains the Test Session ID to which the message pertains.

[R58] The Test Session ID **MUST** be in the range 1 to 2^{32} - 1.

10.2.8 Optional TLVs

SCMs may contain Optional TLVs. Only the SAT TLV, Organization Specific TLVs, and End TLV are expected as an Optional TLV.

[R59] The SAT TLVs **MUST** be included in an Initiate Session Request Message, as defined in section 10.4.

[R60] The SAT TLV **MUST NOT** be included in any SCM except the Initiate Session Request Message.

The general format of an Optional TLV is shown in the figure below:

Type (1 octet)	Length (2 octets)	Value (0 – 65535 octets)

Figure 34 Optional TLV Format

These fields are described further below.

[R61] The first TLV in the SCM if any, **MUST** start at the offset identified by the TLV Offset field in the SCM.

[R62] Each subsequent TLV **MUST** start immediately after the end of the previous TLV.

10.2.8.1 TLV Type

The Type field indicates the type of the TLV. If the value is 0, there are no Length or Value fields. The table below indicates the valid types:

Type	**Meaning**
0	End TLV
38	SAT TLV
31	Organization Specific
1-30, 32-37, 39-255	Out of Scope for this Document

Table 5 SAT Control Protocol SCM TLV Types

[R63] A TLV in an SCM transmitted by a MEP **MUST** have a type value that is not Out of Scope for this Document.

[R64] If an SCM received by a MEP contains a TLV with a type that is Out of Scope for this Document, the TLV **MUST** be copied unmodified into the corresponding SCR, and otherwise ignored.

[R65] If the end of the SCM PDU (including any TLVs) is not immediately followed by the first octet of the FCS, the SCM **MUST** contain an End TLV.

[R66] If the SCM contains an End TLV, it **MUST** be the last TLV in the SCM.

[R67] In the End TLV the Length and Value fields **MUST** not be included.

[R68] If an Organization Specific TLV with an unrecognized OUI or TLV sub-Type is received, it **MUST** be copied unmodified into the corresponding SCR, and otherwise ignored.

10.2.8.2 TLV Length

The Length field indicates the length, in octets, of the Value field. If the Length is 0, there is no Value field.

[R69] The Length in an SCM **MUST NOT** indicate that the TLV Value extends beyond the end of the Ethernet Frame data.

10.2.8.3 TLV Value

The Value field contains data that depends on the Type.

10.3 SAT Control Protocol Response

The SAT Control Protocol Response (SCR) is used by the Responder End to respond to a request from the Controller End or to notify the Controller End about events that have occurred at the Responder End. The SCR is defined below.

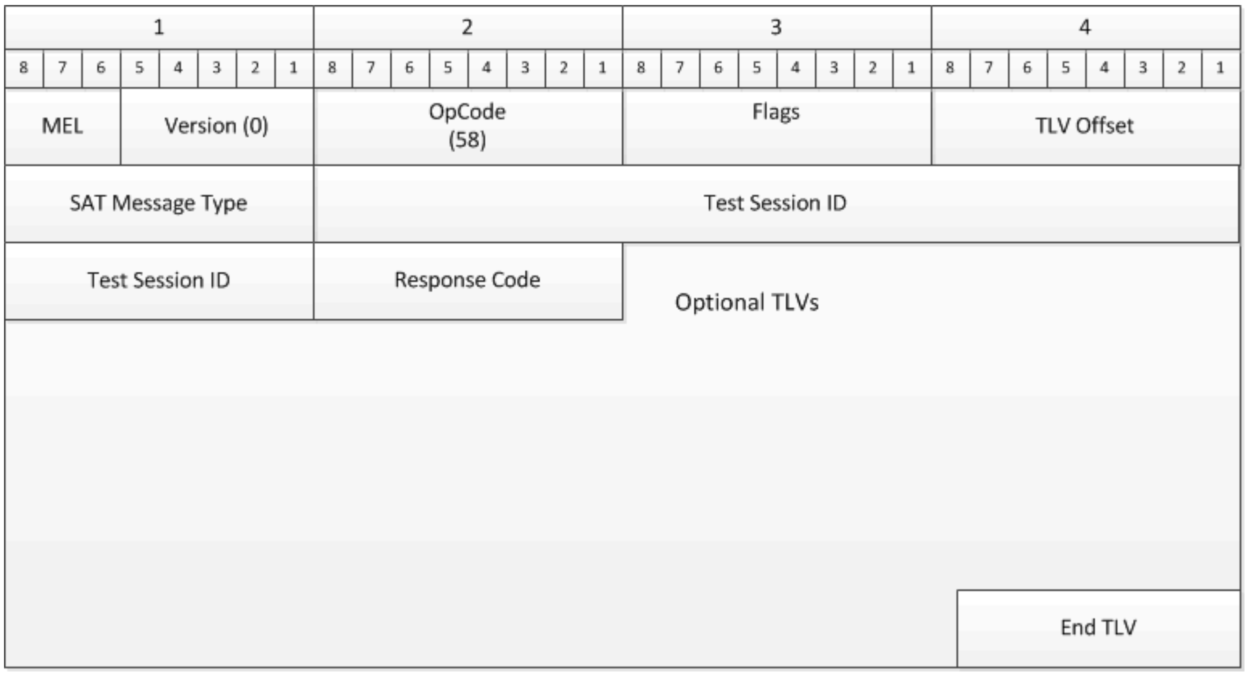


Figure 35 Generic SAT Control Protocol Response Format

[R70] Fixed fields in SCRs **MUST** be located as shown in Figure 35.

10.3.1 MEL

The 3-bit MEL field contains the MEG Level of the MEP to which the message is directed.

[R71] The MEL **MUST** be set to the MEG Level used for SAT.

10.3.2 Version

The 5-bit Version field contains the version of the SCR.

[R72] The Version **MUST** be set to 0 in transmitted SCR frames.

[R73] A MEP **MUST** process and not discard the PDU according to the Version in the PDU or the Version supported by the MEP, whichever is lower, as described in G.8013/Y.1731 [3] clauses 11.2 and 11.3.

10.3.3 OpCode

The value used in the OpCode field is within the range allocated in ITU-T G.8013/Y.1731 [3] for use by MEF.

[R74] The OpCode for the SCR **MUST** be set to 58.

10.3.4 Flags

The Flags field is unused.

[R75] The Flags field **MUST** be set to 0 on transmit.

[R76] The Flags field **MUST** be ignored on receive.

10.3.5 TLV Offset

The TLV Offset indicates the number of octets from this field at which the TLV fields, if any, begin.

[R77] The TLV Offset **MUST** be set to indicate the start of the first TLV.

[R78] The transmitted TLV Offset **MUST** be 6.

[R79] A received SCR **MUST** have a TLV Offset of 6 or greater.

[R80] The TLV Offset **MUST** either indicate a location for the first TLV that is within the SCR PDU, or must indicate a location that corresponds to the first octet of the FCS if this immediately follows the SCR PDU in the frame.

Note: In the latter case, where the TLV Offset indicates a location that corresponds to the first octet of the FCS, then there are no TLVs (including no END TLV).

10.3.6 Message Type

The values for the Message Type are shown in Table 6.

Message Type	Message Name	Description
1	Initiate Session Response Message	Indicates if Responder End accepts test session request or if requested parameters cannot be supported by Responder End
2	Start Session Response Message	Indicates the Responder End has started the test session
3	Stop Session Response Message	Indicates that the Responder End has stopped a test session
4	Abort Session Response Message	Indicates that the Responder End has aborted an in-progress test session.
5	Get Session Status Response Message	Responder End returns status of a test session
6	Fetch Session Results Response Mes-sage	Test session results are returned to the Controller End by the Responder End
7	Delete Session Response Message	Acknowledges that the Responder End has deleted the test session

Table 6 SAT Control Protocol SCR Message Types

[R81] The Message Type in the SCR **MUST** be set as defined in Table 6.

[R82] All other values are reserved and SCRs containing them **MUST** be ignored.

10.3.7 Test Session ID

The Test Session ID field contains the Test Session ID to which the message pertains.

[R83] The Test Session ID **MUST** be in the range 1 to 2^{32} - 1.

10.3.8 Response Code

The Response Code field indicates the error or status in a response. Response codes can be in-cluded in any of the responses. Possible values for SAT Control Protocol are shown in Table 7.

Response Code	Message Name	Symbolic Name	Description
0	No Error	NO_ERROR	No error was contained in the corresponding SCM.

Response Code	Message Name	Symbolic Name	Description
1	Malformed Request	MALFORMED_RQ	The corresponding SCM was not formatted per this document.
2	No Such Test Session	NO_SUCH_SESSION	The specified test session does not exist at the Re-sponder End.
3	Unable to Support	UNABLE_TO_SUPPORT	The Responder End cannot support one or more of the requested Test Session at-tributes.
4	Resource Temporarily Unavailable	TEMP_UNAVAILABLE	The Responder End does not have resources available to support the test session attributes. The attributes can be supported when a re-source becomes available.
5	Test Session Aborted by Admin	ABORTED_BY_ADMIN	The test session was aborted due to the SAT Re-sponder End being adminis-tratively disabled.
6	Test Session Already Exists	SESSION_EXISTS	A test session with that test session ID already exists at the Responder End.
7	Test Session Aborted Due to Fault	ABORTED_DUE_TO_FA ULT	A fault occurred at the Re-sponder End causing the test session to be aborted.
8	Test Session Timed Out	TIMED_OUT	The Responder End did not receive an SCM within specified timeframe and has ended the test session.
9	Unexpected SCM	UNEXP_SCM	The Responder End has re-ceived a SCM that would be a valid message but is not valid due to the current state of the test session.
10-255	Reserved		Reserved

Table 7 MEP Control Message Response Codes

[R84] When the Responder End cannot meet the requested parameters of a test session contained in an Initiate Session Request Message, it **MUST** set the response code set to 3 (UNABLE_TO_SUPPORT) in the Initiate Session Response Message, and copy the SAT TLV(s) specifying the attributes that cannot be supported from the Initiate Session Request Message to the Initiate Session Response Message.

[O3] When the Responder End cannot meet the requested parameters of a test session contained in an Initiate Session Request Message, it **MAY** include an additional copy of the SAT TLV(s) specifying the attributes that cannot be supported in the Initiate Session Response Message, in which the TLV Value has been changed to the nearest value to the requested value that the Responder End can support.

This allows the Responder End to indicate to the Controller End what the closest value that can be supported is. The Controller End can use this information to attempt to initiate a new session using the supported value.

[R85] The Response Code **MUST** be filled with an unreserved value as defined in Table 7.

[R86] SCRs received with a reserved value for the Response Code **MUST** be handled as if they indicate a permanent error condition.

[R87] The Response Code in a Fetch Session Results Response message **MUST** be set to 0 (NO_ERROR).

[R88] The Response Code in a Delete Session Response message **MUST** be set to 0 (NO_ERROR).

The State Machines address the condition if a Fetch Session Results Response Message is received for a session that does not exist. For this reason, the action is not specified in this section.

10.3.9 Optional TLVs

SCRs may contain Optional TLVs. Only the SAT TLV, Organization Specific TLVs, and End TLV are expected as an Optional TLV.

[R89] SAT TLVs **MUST** be included in an Initiate Session Response Message, Get Session Status Response Message or Fetch Session Results Response Message, as described in section 10.4.

[R90] The SAT TLV **MUST NOT** be included in any SCR except the Initiate Session Response Message, Session Status Response Message or Session Results Response Message.

The general format of an Optional TLV is shown in the figure below:

Type (1 octet)	Length (2 octets)	Value (0 – 65535 octets)

Figure 36 Optional TLV Format

These fields are described further below.

[R91] The first TLV in the SCR if any, **MUST** start at the offset identified by the First TLV Offset field in the SCR.

[R92] Each subsequent TLV **MUST** start immediately after the end of the previous TLV.

10.3.9.1 TLV Type

The Type field indicates the type of the TLV. If the value is 0, there are no Length or Value fields. The table below indicates the valid types:

Type	Meaning
0	End TLV
38	SAT TLV
31	Organization Specific
1-30, 32-37, 39-255	Out of Scope for this Document

Table 8 SCR Optional TLV Types

[R93] A TLV in an SCR transmitted by a MEP **MUST** have a type value that is not Out of Scope for this Document, except for responses covered by [R64].

[R94] If an SCR received by a MEP contains a TLV with a type that is Out of Scope for this Document, the TLV **MUST** be ignored.

[R95] If the end of the SCR PDU (including any TLVs) is not immediately followed by the first octet of the FCS, the SCR **MUST** contain an End TLV.

[R96] If the SCR contains an End TLV, it **MUST** be the last TLV in the SCR.

[R97] In the End TLV the Length and Value fields **MUST** not be included

[R98] If an Organization Specific TLV with an unrecognized OUI or TLV sub-Type is received, it **MUST** be ignored.

10.3.9.2 TLV Length

The Length field indicates the length, in octets, of the Value field. If the Length is 0, there is no Value field.

[R99] The Length in an SCR **MUST NOT** indicate that the TLV Value extends beyond the end of the Ethernet Frame data.

10.3.9.3 TLV Value

The Value field contains data that depends on the Type.

10.4 SAT TLVs

The SAT TLV is used to specify test session attributes and test session results. The sub-type field is used to indicate the specific attribute or result contained in the SAT TLV.

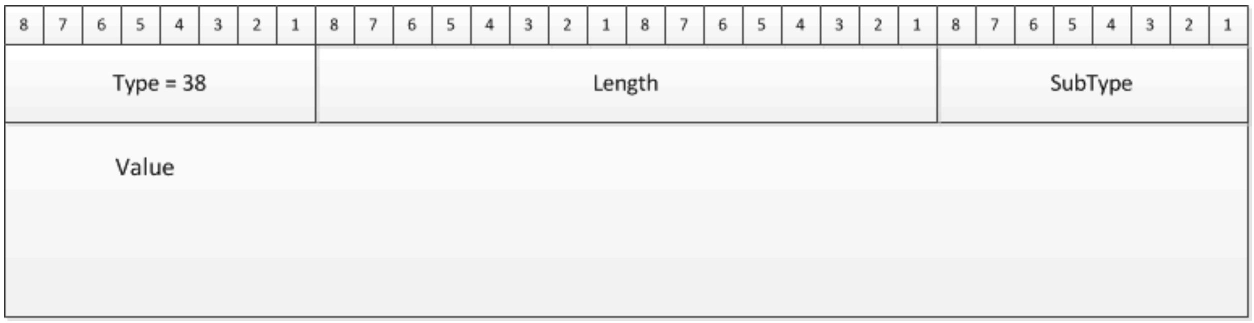


Figure 37 SAT TLV Format

[R100] The SAT TLV **MUST** be encoded as described in IEEE 802.1Q [1] and ITU-T G.803/Y.1731 [3].

[R101] The SAT TLV **MUST** use a Type of 38.

[R102] SubTypes not specified in Table 9 are reserved and **MUST** be ignored on receipt.

[R103] A SAT TLV **MUST** contain only one SubType.

[O4] A SAT PDU **MAY** contain more than one SAT TLV

[R104] A SAT PDU **MUST NOT** contain more than one SAT TLV with the same subtype except in the case of an Initiate Session Response Message with a response code set to 3 (UNABLE_TO_SUPPORT).

[R105] A SAT PDU containing an Initiate Session Response Message with a response code set to 3 (UNABLE_TO_SUPPORT) **MUST NOT** contain more than two SAT TLVs with the same subtype.

An Initiate Session Response Message with a response code set to 3 (UNABLE_TO_SUPPORT) contains the SAT TLV from the Initiate Session Request Message containing the value that cannot be supported, and could contain an additional SAT TLV with the same subtype containing the closest value that can be supported; see [O3].

[R106] If a SAT PDU is received containing a SAT TLV with a SubType that is not valid for that message type (as specified in section 10.4.1), the message **MUST** be discarded.

[O5] Where multiple SAT TLVs are used in a single SCM or SCR, the SAT TLVs **MAY** be placed in any order within the PDU.

SubType Value	TLV Name	Valid Values	Length of Value	Description
0	Measurement Type	0 = FLR only 1 = FLR and Rate	1 Octet	Indicates whether only FLR, or both FLR and Rate (IR or ULR), should be measured
1	MAC Address	Any valid unicast MAC address	6 Octets	GTF or CTF MAC address
2	Destination MAC Address	Any valid MAC address	6 Octets	Destination MAC address
3	Green PCP	0 - 7	1 Octet	Used to indicate Green PCP
4	Yellow PCP	0 - 7	1 Octet	Used to indicate Yellow PCP
5	Duration	1 - 86400	4 Octets	Duration of the test session in seconds
6	EtherType	0x0600 – 0xFFFF	2 Octets	EtherType to be generated
7	SAP	0x0000 – 0xFFFF	2 Octets	DSAP/SSAP to be generated
8	Frame Length	See section 10.4.2.7	See section 10.4.2.7	Length of test frames (containing FL_PDUs), in oc-

SubType Value	TLV Name	Valid Values	Length of Value	Description
9	Frame Pattern	0 = Repeating Pattern 1 = PRBS31 See section 10.4.2.8	1 Octet or 9 Octets – see section 10.4.2.8	Either PRBS31 or 8 Octet repeating pattern
10	Frame Quantity	0 - 2^{64}-1	8 octets	Number of frames to be transmitted or number of green frames received
11	Frame Interval	1 - 65535	2 Octets	The time between transmitted frames in milliseconds
12	Green Rate	1 – 2^{32}-1	4 Octets	Rate of Green frames to transmit in kbps
13	Yellow Rate	1 – 2^{32}-1	4 Octets	Rate of Yellow frames to transmit in kbps
14	Yellow Frame Quantity	0 - 2^{64}-1	8 Octets	Number of Yellow frames transmitted or received
15	Measured Rate Duration	1 - 2^{64}-1	8 Octets	Time in nanoseconds from the transmission or receipt of the first time stamped frame to the transmission or receipt of the last time stamped frame in the test session
16	Test Session Status	See Table 12	1 Octet	Test Session status

SubType Value	TLV Name	Valid Values	Length of Value	Description
17	Yellow DEI	0 = DEI unset 1 = DEI set	1 Octet	DEI value for Yel-low frames
18	Rate Type	0 = IR 1 = ULR	1 Octet	Whether frame rates are expressed as IR or ULR
19	Measured Rate Green Bits	$0 – 2^{64}$-1	8 Octets	Number of Green bits transmitted or received between the first time stamped frame and the last time stamped frame in the test session
20	Measured Rate Yel-low Bits	$0 – 2^{64}$-1	8 Octets	Number of Yellow bits transmitted or received between the first time stamped frame and the last time

Table 9 TLV SubTypes

The "TLV Name" column in Table 9 indicates how a SAT TLV with the given subtype is re-ferred to in the following Sections; for instance, a SAT TLV with the subtype set to 5 is referred to as a "Duration TLV".

[R107] Values for the SAT TLV subtypes **MUST** be within the ranges specified in Ta-ble 9.

The Length column in Table 9 indicates the length of the Value field, not including the SubType. Therefore the Length field in the TLV must be at least 1 greater than this.

[R108] The length field in a transmitted SAT TLV **MUST** be exactly one greater than the length given in Table 9.

[R109] The length field in a received SAT TLV **MUST** be at least one greater than the length given in Table 9.

10.4.1 Tests and TLV Type Use

Test to be performed as a part of SAT are defined in the SAT Technical Specification [12]. This sub-section defines for each test which subtype values are included in the Initiate Session Request Message, Initiate Session Response Message, the Get Session Status Response Message and the Fetch Session Results Response Message. For this purpose, the tests defined in the SAT Technical Specification [12] are categorized as follows:

- L2CP Tests
    - [None specified in SAT TS]
- Frame Delivery Tests:
    - OVC MTU Size Test [SAT TS section 10.3.1]
    - CE-VLAN ID Preservation Test [SAT TS section 10.3.2]
    - CE-VLAN CoS ID Preservation Test [SAT TS section 10.3.3]
    - Broadcast, Unicast, Multicast Frame Delivery Test [SAT TS section 10.3.4]
- Bandwidth Tests:
    - Ingress Bandwidth Profile per OVC End Point Tests [SAT TS section 10.3.5]
    - Service Performance Tests [SAT TS section 10.4]

Note: At this time burst testing is outside the scope of SAT. Mechanisms to support burst testing are not addressed in this document. This may be addressed in a later phase of the project.

[R110] An Initiate Session Request Message **MUST** include a valid set of SAT TLVs as indicated in Table 10.

Note: “F” indicates that the TLV subtype is required for a Forward test, “B” indicates it is required for a Backward test. “(F)” or “(B)” indicate the TLV subtype is optional in a Forward or Backward test respectively; in this case there are further requirements in the following Sections.

[R111] An Initiate Session Request Message received with a set of SAT TLVs not indicated in Table 10 is invalid and **MUST** be discarded.

Subtypes	**L2CP Test**	**Frame Delivery Test - Frame Count Specified**	**Frame Delivery Test - Rate Specified**	**Bandwidth Test**
Measurement Type	F, B	F, B	F, B	F, B
MAC Address	F	F	F	F

Subtypes	**L2CP Test**	**Frame Delivery Test - Frame Count Specified**	**Frame Delivery Test - Rate Specified**	**Bandwidth Test**
Destination MAC Address	(F), B	(F), B	(F), B	B
Green PCP	F, B	F, B	F, B	F, B
Yellow PCP				(F), (B)
Duration	F	F	F, B	F, B
EtherType	(B)			
SAP	(B)			
Frame Length		(B)	(B)	(B)
Frame Pattern		(B)	(B)	(B)
Frame Quantity	B	B		
Frame Interval	B	B		
Green Rate			B	B
Yellow Rate				(B)
Yellow DEI				(F), (B)
Rate Type			B	F, B

Table 10 Initiate Session Request Message TLVs

[R112] An Initiate Session Response Message with the Response Code set to 0 (NO_ERROR) **MUST** contain a MAC address TLV, except in the case of a Forward test where the Session Initiate Request message included a Destination MAC Address TLV containing a broadcast or multicast address.

[R113] An Initiate Session Response Message with the Response Code set to 0 (NO_ERROR) **MUST NOT** contain a MAC address TLV in the case of a Forward test where the Session Initiate Request message included a Destination MAC Address TLV containing a broadcast or multicast address.

[R114] An Initiate Session Response Message with the Response Code set to 0 (NO_ERROR) **MUST NOT** contain any TLV other than a MAC Address TLV.

[R115] An Initiate Session Response Message with the Response Code set to a value other than 0 (NO_ERROR) or 3 (UNABLE_TO_SUPPORT) **MUST NOT** contain any SAT TLVs.

[R116] A Session Result Response Message **MUST** include the SAT TLVs indicated in Table 11 for the type of test as indicated in the corresponding Initiate Session Request Message.

[R117] A Session Result Response Message received with an invalid set of SAT TLVs **MUST** be discarded.

Subtypes	L2CP Test	Frame Delivery Test - Frame Count Specified	Frame Delivery Test - Rate Specified	Bandwidth Test
Frame Quantity	F, B	F, B	F, B	F, B
Yellow Frame Quantity				(F), (B)
Measured Rate Duration				F, B
Measured Rate Green Bits				F, B
Measured Rate Yellow Bits				(F), (B)
Rate Type				F, B

Table 11 Fetch Session Results Response Message TLVs

[R118] A Get Session Status Response Message with the Response Code set to 0 (NO_ERROR) **MUST** contain a Status TLV and no other SAT TLVs.

[R119] A Get Session Status Response Message with the Response Code set to a value other than 0 (NO_ERROR) **MUST NOT** contain any SAT TLVs.

10.4.2 SAT TLV Subtypes

10.4.2.1 Measurement Type

The Measurement Type TLV is used to identify what is to be measured for both Forward and Backward tests.

[R120] The Measurement Type **MUST** be set to 0 (FLR-only) for Frame Delivery and L2CP tests.

[R121] The Measurement Type **MUST** be set to 1 (FLR and Rate) for Bandwidth tests.

Other values for the Measurement Type are reserved.

[R122] If an Initiate Session Request Message is received with a Measurement Type other than 0 or 1, an Initiate Session Response **MUST** be generated with the Response Code set to 3 (UNABLE_TO_SUPPORT).

10.4.2.2 MAC Address and Destination MAC Address

The MAC Address and Destination MAC Address TLVs are used to identify the MAC address of the GTF and CTF, and the destination MAC address for broadcast and multicast frames. They are used in combination as follows:

- For a Forward unicast test, the GTF MAC address is sent in the Initiate Session Request Message, and the CTF MAC address is returned in the Initiate Session Response Message. The GTF at the Controller End uses its own MAC address as the source address and the received CTF address as the destination address. The CTF at the Responder End uses the received GTF MAC Address as the source address and its own address as the destination address.
- For a Forward multicast or broadcast test, both the GTF MAC address and the desired destination MAC are sent in the Initiate Session Request, and neither of the MAC address TLVs is included in the response. The GTF at the Controller End uses its own address as the source address and the specified address as the destination address. The CTF at the Response End uses the received GTF Address as the source and the received address as the destination address.
- For a Backward unicast test, the MAC address of the CTF at the Controller End is sent as the destination in the Initiate Session Request Message, and the MAC address of the GTF at the Responder End is returned in the Initiate Session Response Message. The GTF at the Responder End uses its own address as the source address and the received destination address. The CTF at the Controller End uses the received GTF address as the source address and its own address as the destination address.
- For a Backward multicast or broadcast test, the desired destination MAC address is sent in the Initiate Session Request Message, and the MAC address of the GTF at the Responder End is returned in the Initiate Session Response Message. The GTF at the Responder End uses its own address as the source address and the received address as the destination address. The CTF at the Controller End uses the received GTF address as the source address and the specified multicast or broadcast address as the destination address.

[R123] In an Initiate Session Request Message for a Forward test, a MAC Address TLV **MUST** be included containing the unicast address of the GTF at the Controller End.

[O6] An Initiate Session Request Message for a Forward Test **MAY** include a Destination MAC Address TLV containing a valid multicast or broadcast address.

[R124] An Initiate Session Request Message for a Forward Test **MUST NOT** include a Destination MAC Address TLV containing a unicast address.

[R125] In an Initiate Session Response Message for a Forward test, when a MAC Address TLV is included (see [R112] and [R113]), it **MUST** contain the unicast MAC address of the CTF at the Responder End.

[R126] For a Forward test, the GTF at the Controller End **MUST** use its own MAC address as the source MAC address for generated test frames.

[R127] For a Forward test, if a Destination MAC address TLV containing a broadcast or multicast address was included in the Initiate Session Request Message, the GTF at the Controller End **MUST** use this address as the destination address for generated test frames.

[R128] For a Forward test, if no Destination MAC address TLV was included in the Initiate Session Request Message, the GTF **MUST** use the address contained in the MAC address TLV in the Initiate Session Response Message as the destination address for generated test frames.

[R129] For a Forward test, the CTF at the Responder End **MUST** use the address contained in the MAC address TLV in the Initiate Session Request Message as the source address when determining frames to count.

[R130] For a Forward test, if a Destination MAC address TLV containing a broadcast or multicast address was included in the Initiate Session Request Message, the CTF at the Responder End **MUST** use this address as the destination address when determining frames to count.

[R131] For a Forward test, if no Destination MAC address TLV was included in the Initiate Session Request Message, the CTF at the Responder End **MUST** use its own address as the destination address when determining frames to count.

[R132] In an Initiate Session Response Message for a Backward test, the MAC Address TLV **MUST** contain the unicast MAC address of the GTF at the Responder End.

[R133] An Initiate Session Request Message for a Backward test **MUST** include a Destination MAC Address TLV containing either the unicast address of the CTF at the Controller End, or a valid multicast or broadcast address.

[R134] For a Backward test, the GTF at the Responder End **MUST** use its own MAC address as the source MAC address for generated test frames.

[R135] For a Backward test, the GTF at the Responder End **MUST** use the address contained in the Destination MAC Address TLV as the destination MAC address for generated test frames.

[R136] For a Backward test, the CTF at the Controller End **MUST** use the address contained in the MAC address TLV in the Initiate Session Response Message as the source address when determining frames to count.

[R137] For a Backward test, if the Destination MAC address TLV included in the Initiate Session Request Message contained a broadcast or multicast address, the CTF at the Controller End **MUST** use this address as the destination address when determining frames to count.

[R138] For a Backward test, if the Destination MAC address TLV included in the Initiate Session Request Message contained a unicast address, the CTF at the Controller End **MUST** use its own address as the destination address when determining frames to count.

10.4.2.3 Green PCP

The Green PCP TLV is used to identify the value used to indicate green frames. This is used by both GTFs and CTFs at the Responder End.

[R139] The GTF for the session **MUST** transmit green frames using the specified green PCP value and a DEI value of 0.

[R140] The CTF for the session **MUST** identify green frames to be counted using the specified green PCP value and DEI value of 0.

Note: If multiple PCP values are used to indicate green frames, separate test sessions are run for each value.

10.4.2.4 Yellow PCP

The Yellow PCP TLV is used to identify the value used to indicate Yellow frames. This is used by both GTFs and CTFs at the Responder End during Bandwidth test sessions.

[O7] An Initiate Session Request Message for a bandwidth test **MAY** include a Yellow PCP TLV.

[R141] If an Initiate Session Request Message for a bandwidth test includes a Yellow PCP TLV, the CTF for the session **MUST** maintain separate counts of frames and octets for green and yellow frames, and identify yellow frames to be counted using the specified yellow PCP value, along with the yellow DEI value of [R184].

[R142] If an Initiate Session Request Message for a Forward bandwidth test includes a Yellow PCP TLV, the GTF at the Controller End **MUST** generate yellow test frames with the indicated value.

[R143] If an Initiate Session Request Message for a Backward bandwidth test includes a Yellow PCP TLV, the GTF at the Responder End **MUST** generate yellow test frames with the indicated value.

[R144] An Initiate Session Request Message for a bandwidth test in the Backward direction **MUST** include a Yellow PCP TLV if and only if it includes a Yellow IR TLV.

Note 1: A Yellow DEI TLV is included if and only if a Yellow PCP TLV is included, see section 10.4.2.16.

Note 2: If multiple PCP values are used to indicate yellow frames, separate test sessions are run for each value.

10.4.2.5 Duration

The Duration TLV is used to indicate the desired or expected duration in seconds of the test session. The maximum Duration of a test session is 86400 seconds (1 day).

[R145] An Initiate Session Request Message for a Forward test **MUST** include a Duration TLV containing the expected duration of the test; that is, the time between the transmission of the first test frame and the last test frame.

Note that this is used in the state machine to set the wait timer, as specified in section 9.3.

[R146] An Initiate Session Request Message for a Bandwidth test in the Backward direction **MUST** include a Duration TLV containing the length of time for which the GTF at the Responder End should generate frames.

[O8] An Initiate Session Request Message for a Frame Delivery test in the Backward direction **MAY** include a Duration TLV containing the length of time for which the GTF at the Responder End should generate frames.

Note: As shown in Table 10, an Initiate Session Request Message for a test in the Backward direction includes either a Duration TLV and a Green Rate TLV, or a Frame Quantity TLV and a Frame Interval TLV, but not any other combination.

[R147] If an Initiate Session Request Message for a test in the Backward direction includes a Duration TLV, the GTF at the Responder End **MUST** generate frames for the duration specified in the Duration TLV.

10.4.2.6 EtherType and SAP

The EtherType or SAP TLV is used to identify the EtherType or SAP for frames generated by the GTF at the Responder End, for L2CP tests. In the case of a SAP TLV, the 4-octet value contains the DSAP in the first (high-order) two octets and the SSAP in the second (low-order) two octets.

[O9] An Initiate Session Request Message for a Backward test **MAY** include an EtherType TLV or a SAP TLV.

[R148] An Initiate Session Request Message for a Backward test **MUST NOT** include both an EtherType TLV and a SAP TLV.

[R149] If the Initiate Session Request Message for a Backward test contains an EtherType TLV, the GTF at the Responder End **MUST** generate type-encoded frames with the specified EtherType.

[R150] If the Initiate Session Request Message for a Backward test contains a SAP TLV, the GTF at the Responder End **MUST** generate length-encoded frames with the specified DSAP and SSAP.

[R151] If the Initiate Session Request Message for a Backward test does not contain an EtherType TLV or a SAP TLV, the GTF at the Responder End **MUST** generate FL-PDU frames as specified in section 8.1.

10.4.2.7 Frame Length

The Frame Length TLV is used to indicate the length of a frame in octets to be transmitted by the GTF at the Responder End of a Backward test session. The frame length includes the header to the FCS and excludes the preamble and inter-packet gap. When frames of different lengths are to be transmitted such as when using an EMIX pattern, multiple frame lengths are included in a single SAT TLV. The GTF at the Responder End of a Backward test session transmits frames of the lengths indicated in a circular manner repeating for the duration of the test session.

The TLV length is used to identify the number of lengths, each represented by two octets. The TLV length also includes the one-octet subtype field; hence, a TLV length of three indicates one Frame Length. A TLV length of 65 indicates 32 Frame lengths.

[O10] An Initiate Session Request Message for a Backward session MAY include a Frame Length TLV.

[R152] An Initiate Session Request Message for a Backward session **MUST NOT** include a Frame Length TLV if it includes an EtherType or SAP TLV.

[R153] A Frame Length TLV **MUST** have a length between 3 and 65 octets, in 2-octet increments.

[R154] Each frame length **MUST** be encoded as a 2-octet field containing the desired frame length in octets.

[R155] The first frame length field **MUST** immediately follow the subtype field.

[R156] Each subsequent frame length field **MUST** immediately follow the preceding one.

[R157] Each frame length specified in the Frame Length TLV **MUST** be in the range 64-9600.

[R158] A frame length of 64-1526 bytes **MUST** be supported.

[D1] A frame length of 1527-2000 bytes **SHOULD** be supported.

[O11] A frame length of 2001-9600 bytes **MAY** be supported.

Note: IEEE defines a valid Ethernet frame to be 64-2000 bytes. That is why anything larger is defined as an optional requirement.

[R159] An implementation of a GTF **MUST** support generating frames of up to 32 frame lengths, in a round-robin fashion.

[R160] If an Initiate Session Request Message includes a Frame Length TLV, the GTF at the Responder End **MUST** transmit a frame of each length contained in the TLV, in order, and repeat that process until all requested frames have been transmitted.

[R161] If an Initiate Session Request Message does not include a Frame Length TLV, the GTF at the Responder End **MUST** transmit frames of length 64 octets.

Note: A Frame Length TLV can include the same frame length more than once; for example, to generate 75% 64-octet frames and 25% 1500-octet frames, a frame length TLV could be included containing 4 frame length values, of 64, 64, 64, and 1500.

10.4.2.8 Frame Pattern

The Frame Pattern TLV is used to indicate to the GTF at the Responder End of a Backward test session the pattern of any stuff bits contained in a transmitted frame.

[O12] An Initiate Session Request Message for a Backward session MAY include a Frame Pattern TLV.

A Frame Pattern TLV consists of a one-octet type field indicating whether the pattern should be a Pseudo-Random Bit Sequence (PRBS31) or a repeating 8-octet pattern. In the latter case, the type is followed by the desired 8-octet pattern. Taking into account the one-octet subtype field, the TLV length is thus either 2 octets or 10 octets.

[R162] A Frame Pattern TLV **MUST** include a one octet type field immediately after the subtype.

[R163] The type field **MUST** be set either to 0, indicating an 8-octet repeating pattern, or 1 indicating a (PRBS31).

Other values for the type are reserved.

[R164] If a Frame Pattern TLV is received containing a type other than 0 or 1, the TLV **MUST** be ignored.

[R165] A Frame Pattern TLV with type 0 **MUST** have TLV length set to 10.

[R166] A Frame Pattern TLV with type 1 **MUST** have TLV length set to 2.

[R167] A Frame Pattern TLV with type 0 **MUST** include the desired 8-octet pattern immediately after the type field.

[R168] If an Initiate Session Request Message for a Backward session includes a Frame Pattern TLV, the GTF at the Responder End **MUST** generate FL-PDUs containing a Data TLV with the TLV length necessary to achieve the required frame length.

[R169] If an Initiate Session Request Message for a Backward session includes a Frame Pattern TLV with type 0, the Data TLV **MUST** contain the specified 8-octet pattern, repeated as many times as necessary to fill the TLV. The last repetition may be truncated if the required length is not a multiple of 8.

[R170] If an Initiate Session Request Message for a Backward session includes a Frame Pattern TLV with type 1, the Data TLV **MUST** contain a PRBS31 pattern of the required length, as specified in 8.1.7.4.

[R171] If an Initiate Session Request Message for a Backward session does not includes a Frame Pattern TLV, the GTF at the Responder End **MUST** generate FL-PDUs that do not contain a Data TLV, and with sufficient padding after the END TLV to achieve the required frame length.

10.4.2.9 Frame Quantity

The Frame Quantity TLV is used to identify the number of frames transmitted or to be transmitted by the GTF at the Responder End of a Backward test session or the number of green frames received by the CTF at the Responder End of a Forward test session.

[R172] If an Initiate Session Request Message for a Backward test includes a Frame Quantity TLV, the GTF at the Responder End **MUST** generate the number of frames specified in the Frame Quantity TLV in the Initiate Session Request Message.

Note: As shown in Table 10, an Initiate Session Request Message for a test in the Backward direction includes either a Duration TLV and a Green Rate TLV, or a Frame Quantity TLV and a Frame Interval TLV, but not any other combination.

[R173] The Frame Quantity TLV **MUST** be used in a Session Results Response Message to indicate the number of green frames received by the CTF at the Responder End of a Forward test session.

[R174] The Frame Quantity TLV **MUST** be used in a Session Results Response Message to indicate the number of green frames transmitted by the GTF at the Responder End of a Backward test session.

10.4.2.10 Frame Interval

The Frame Interval TLV is used to specify the interval, in milliseconds, between frames to be transmitted by the GTF at the Responder End of a Backward OVC Data Service Frame Count Specified or L2CP test session.

[R175] If an Initiate Session Request Message for a Backward test includes a Frame Interval TLV, the GTF at the Responder End **MUST** generate frames at the interval specified in the Frame Interval TLV in the Initiate Session Request Message.

10.4.2.11 Green Rate

The Green Rate TLV is used to identify the IR or ULR of green frames in kb/s that are to be transmitted by the GTF at the Responder End of a Backward Frame Delivery or Bandwidth test session. IR or ULR is specified using the Rate Type TLV (see 10.4.2.17).

[R176] If an Initiate Session Request Message for a Backward test includes a Green Rate TLV, the GTF at the Responder End **MUST** generate green frames at the IR or ULR (as indicated in the Rate Type TLV) specified in the Green Rate TLV in the Initiate Session Request Message.

Note: When frames of different lengths are being sent, the average rate is to be as requested; however the exact method by which this is achieved is implementation specific.

10.4.2.12 Yellow Rate

The Yellow Rate TLV is used to identify the IR or ULR of yellow frames in kb/s that are to be transmitted by the GTF at the Responder End of a Backward Bandwidth test session. IR or ULR is specified using the Rate Type TLV (see 10.4.2.17).

[O13] For a Backward Bandwidth test, the Initiate Session Request Message **MAY** include a Yellow Rate TLV.

Note: In this case the Initiate Session Request Message is required also to contain a Yellow PCP TLV, see [R144] and [R184].

[R177] If the Initiate Session Request Message for a Backward Bandwidth test includes a Yellow Rate TLV, the GTF at the Responder End **MUST** generate yellow frames at the specified IR or ULR (as indicated in the Rate Type TLV) concurrently with the generation of Green frames as specified in [R176].

Note: When frames of different lengths are being sent, the average rate is to be as requested; however the exact method by which this is achieved is implementation specific.

10.4.2.13 Yellow Frame Quantity

The Yellow Frame Quantity TLV is used to report the number of yellow frames transmitted by the GTF at the Responder End of a Backward test session or the number of yellow frames received by the CTF at the Responder End of a Forward test session.

[R178] If an Initiate Session Request Message for a Forward session included a Yellow PCP TLV, the Fetch Session Results Response Message **MUST** include a Yellow Frame Quantity TLV containing the number of yellow frames received by the CTF at the Responder End.

[R179] If an Initiate Session Request Message for a Backward session included a Yellow PCP TLV, the Fetch Session Results Response Message **MUST** include a Yellow Frame Quantity TLV containing the number of yellow frames transmitted by the GTF at the Responder End.

10.4.2.14 Measured Rate Duration

The Measured Rate Duration TLV is used by the Responder End when IR/ULR is to be measured, to indicate the time, in nanoseconds, from when the first bit of the MAC DA of the first frame for the session is transmitted or the last bit of the FCS of the first frame is received until the first bit of the MAC DA of the last frame for the session is transmitted or the last bit of the FCS of the last frame is received. This does not use a timestamp (e.g., a 1588v1 timestamp), but is a count of nanoseconds. This is used by the Controller End to determine the IR or ULR that was observed at the Responder End.

[R180] For a Forward Bandwidth test, the Fetch Session Results Response Message **MUST** include a Measured Rate Duration TLV containing the time, in nanoseconds, between the first test frame for the session that is included in the bit counts being received, and the last frame for the session that is included in the bit counts being received.

[R181] For a Backward Bandwidth test, the Fetch Session Results Response Message **MUST** include a Measured Rate Duration TLV containing the time, in nanoseconds, between the first test frame for the session that is included in the bit counts being transmitted, and the last frame for the session that is included in the bit counts being transmitted.

Note: It might not be possible for implementations to timestamp every transmitted or received frame to the required level of accuracy and precision. It is expected that implementations can support taking the timestamps of a frame near the beginning of the test and another near the end of the test and can use these to calculate the duration; however, in this case frames received before the first timestamp or after the final one are not included in the bit counts reported in the Fetch Session Results Response Message using the Measured Rate Green Bits and Measured

Rate Yellow Bits TLVs. This allows the Controller End to accurately calculate the actual transmitted or received IR or ULR (based on the bit counts and duration), as well as the FLR (based on the Frame and Octet counts).

10.4.2.15 Test Session Status

The Test Session Status TLV is used by the Responder End of a test session to indicate the status of a test session. It is used only in a Get Session Status Response Message.

[R182] Get Session Status Response Messages **MUST** be transmitted with the Test Session Status TLV set to one of the values given in Table 12, as specified by the state machine described in section 9.3.

Values not given in Table 12 are reserved.

[R183] If a Get Session Status Response Message is received containing a Status TLV with a value not given in Table 12, the status **MUST** be treated as unknown.

Status Value	**Symbolic Name**	**Description**
1	NOT_STARTED	The test session has not started.
2	RUNNING	The test session is running.
3	STOPPED	The test session has stopped.
4	DELETE	The test session results have been fetched and the test session is ready to be deleted.

Table 12 Session Status Values

10.4.2.16 Yellow DEI

The Yellow DEI TLV is used to indicate if DEI is used to indicate Yellow frames.

[O14] An Initiate Session Request Message for a bandwidth test **MAY** include a Yellow DEI TLV.

[R184] If an Initiate Session Request Message for a bandwidth test includes a Yellow DEI TLV, the CTF for the session **MUST** use the indicated value to determine yellow frames to count, along with the Yellow PCP value of [R141].

[R185] If an Initiate Session Request Message for a bandwidth test includes a Yellow DEI TLV, the GTF for the session **MUST** generate test frames with the DEI set as indicated.

[R186] An Initiate Session Request Message **MUST** include a Yellow DEI TLV if and only if it includes a Yellow PCP TLV.

Note: An Initiate Session Request Message for a Backward test includes a Yellow DEI TLV if and only if it includes a Yellow IR TLV; this is implied by the combination of [R186] (Yellow DEI if and only if Yellow PCP) and [R144] (Yellow PCP if and only if Yellow IR).

[R187] If an Initiate Session Request Message includes a Yellow DEI TLV with value 0, the value in the Yellow PCP TLV **MUST** be different from the value in the Green PCP TLV.

[D2] If an Initiate Session Request Message includes a Yellow DEI TLV with value 1, the value in the Yellow PCP TLV **SHOULD** be the same as the value in the Green PCP TLV.

Note: The color of a frame is indicated by either a PCP value or DEI value.

10.4.2.17 Rate Type

The Rate Type TLV is used for bandwidth tests, to indicate whether the Green Rate and Yellow Rate TLVs contain values for IR or ULR, and whether the Measured Rate Green Bits and Measured Rate Yellow Bits TLVs contain bit counts that correspond to IR or ULR. The Responder End returns the Rate Type TLV value in the Fetch Session Results Response message to indicate which method, IR or ULR, was used to calculate the Measured Rate Green/Yellow Bits. The Controller End can then convert the result to the desired method as appropriate.

[R188] If an Initiate Session Request Message for a Backward test contains a Green Rate TLV or a Yellow Rate TLV, it **MUST** include a Rate Type TLV indicating whether the rates are expressed as IR (if the Rate Type is 0) or ULR (if the Rate Type is 1).

[O15] If an Initiate Session Request Message includes a Rate Type TLV, the Responder End **MAY** use this to select the rate type used in the Fetch Session Results Response message for Measured Rate Green Bits TLV and (if present) Measured Rate Yellow Bits TLV.

[R189] In a Fetch Session Results Response Message, if the Rate Type TLV is 0, the values in the Measured Rate Green Bits and (if present) the Measured Rate Yellow Bits TLV **MUST** include counts of the number of bits sent or received that correspond to Information Rate; that is, starting with the first MAC address bit and ending with the last FCS bit of each frame.

[R190] In a Fetch Session Results Response Message, if the Rate Type TLV is 1, the values in the Measured Rate Green Bits and (if present) the Measured Rate Yellow Bits TLV **MUST** include counts of the number of bits sent or received that correspond to Utilized Line Rate; that is, for each frame including the bits a) allocable to the minimum-duration period of each Inter-Packet gap (but not the number of bits allocable to the part of each Inter-Packet gap longer than the minimum), b) in the preamble, c) in the start of frame delimiter and d) in the

Ethernet Service Frame starting with the first MAC address bit and ending with the last FCS bit.

Other values for the Rate Type are reserved.

10.4.2.18 Measured Rate Green Bits

The Measured Rate Green Bits TLV is used for bandwidth tests, to indicate the number of bits transmitted or received for green frames. The value corresponds to the IR or the ULR, depending on the value of the Rate Type.

[R191] The count of green bits reported in the Session Results Response Message **MUST** be consistent with the Measured Rate Duration (see section 10.4.2.14).

10.4.2.19 Measured Rate Yellow Bits

The Measured Rate Yellow Bits TLV is used for bandwidth tests, to indicate the number of bits transmitted or received for yellow frames. The value corresponds to the IR or the ULR, depending on the value of the Rate Type.

[R192] If an Initiate Session Request Message for a Forward session included a Yellow PCP TLV, the Fetch Session Results Response Message **MUST** include a Measured Rate Yellow Bits TLV containing the number of bits for yellow frames received by the CTF at the Responder End.

[R193] If an Initiate Session Request Message for a Backward session included a Yellow PCP TLV, the Fetch Session Results Response Message **MUST** include a Measured Rate Yellow Bits TLV containing the number of bits for yellow frames transmitted by the GTF at the Responder End.

[R194] The count of yellow bits reported in the Session Results Response Message **MUST** be consistent with the Measured Rate Duration (see section 10.4.2.14).

11. References

[1] IEEE 802.1Q-2011, Virtual Bridged Local Area Networks, 2011 IEEE Standard for Local and metropolitan area networks–Media Access Control (MAC) Bridges and Virtual Bridge Local Area Networks

[2] IETF RFC 2119, Key Words for use in RFCs to Indicate Requirement Levels, 1997

[3] ITU-T G.8013/Y.1731,OAM Functions and Mechanisms for Ethernet-based Networks, 2013

[4] MEF 4, Metro Ethernet Network Architecture Framework – Part 1: Generic Framework, 2004

[5] MEF 10.3, Ethernet Service Attributes – Phase 3, 2013

[6] MEF 12.2, Carrier Ethernet Network Architecture Framework Part 2: Ethernet Services Layer, 2014

[7] MEF 17, Service OAM Requirements and Framework – Phase 1, 2007

[8] MEF 23.1, Carrier Ethernet Class of Service – Phase 2, 2012

[9] MEF 26.1, External Network Network Interface – Phase 2, 2012

[10] MEF 35, Service OAM Performance Monitoring Implementation Agreement, 2012

[11] ITU-T Y.1564, Ethernet Service Activation Test Methodology, 2011

[12] Carrier Ethernet Service Activation Testing

[13] IEEE 1588-2002, IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems, 2002

[14] MEF 46, Latching Loopback Protocol and Functionality

[15] ITU-T G.8021, Characteristics of Ethernet Transport Network Equipment Functional Blocks, 05/2012

[16] MEF 33, Ethernet Access Services Definition, 2012

[17] IEEE 802-2014 Standard for Local and Metropolitan Area Networks: Overview and Architecture

12. Appendix I Interaction Diagrams

The Interaction Diagrams shown in this Appendix provide an overall view of the messages exchanged between the various components that are specified in the SAT PDU for the purpose of performing configuration and performance measurements as part of the SAT PDU control protocols. The SAT PDU protocols provide the ability to configure and control the SAT steps and to fetch test results at the completion of the test.

12.1 Key

The symbols used in the diagrams are shown below.

Note that the only type of message used in the normative section of this document is the SAT PDU protocol message. The other types of messages are provided for completeness but are not part of the SAT PDU protocols. Those include the messages exchanged between the Ethernet Test Support System (ETSS) and the SAT PDU Controller End as well as between the Controller End MEP and the Responder End MEP to their respective GTF and CTF. Internal messages are messages between the Controller End or Responder End MEP and its GTF or CTF. External messages are messages that pass between the ETSS and Controller End or between the NMS and Responder End MEP.

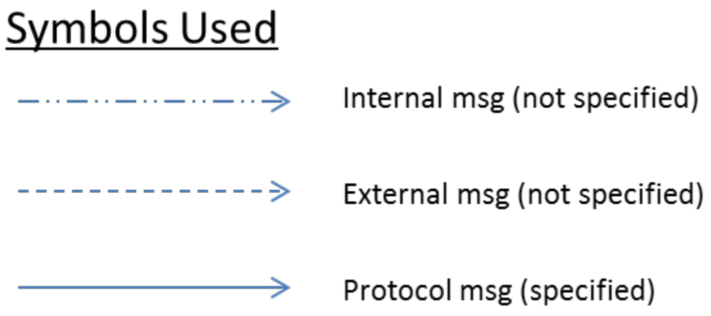


Figure 38 Interactive Drawings Key

12.2 Test Configurations

There are two test configurations used in SAT.

- A Forward test where the GTF is local to the Controller End
- A Backward test where the GTF is remote to the Controller End

Note that except for the ETSS, all the other components are assumed to reside within ETEs. The SAT PDU Controller End MEP and the GTF or CTF may reside on the same device, or may be distributed across devices. Similarly, the SAT PDU Responder End MEP and GTF or CTF may reside on the same device, or may be distributed across devices. The ETSS may reside within the same device as the Controller End MEP or Responder End MEP (e.g. in an ETE-I), or may reside on a different device.

The ETSS acts as the test management system. It has interactions with the test operator, on one hand, and the ETEs and/or MEPs participating in the test, on the other hand.

The SAT PDU Controller End is the component that controls and coordinates the test session sequence and requests the test results from the SAT PDU Responder End MEP at the end of the test session. The Controller End is capable of managing multiple unrelated test sessions in parallel. However, in these interaction diagrams only a single test session is shown.

Finally, at the Responder End MEP there may or may not be an interaction with ETSS, since no tests are conducted from this side. Instead, a management system (NMS) may interact with the Responder End MEP to enable responding to requests from the Controller End and optionally receive notification messages regarding the status of the conducted test session(s).

12.2.1 Forward Test Session

12.2.1.1 Session Initiation

Figure 39 below depicts the session Initiation message interaction for a Forward test.

The ETSS sends an external message request to the Controller End MEP, requesting the initialization of a test session. The Controller End MEP, in turn, initiates a sequence of control messages to create the GTF and CTF.

The Controller End MEP sends an internal local message to create the GTF and sends an Initiate Session Request Message to the Responder End MEP to create the CTF on its side. The Responder End MEP sends an internal local message to create the CTF and sends an Initiate Session Response Message to the Controller End MEP with the results of its operation, and, optionally, sends an external status message to the ETSS/NMS at the Responder End MEP. The Controller End MEP informs the ETSS using an external message of the results of its request. If the session initiation is successful, the result includes a Session ID, generated by the Controller End MEP.

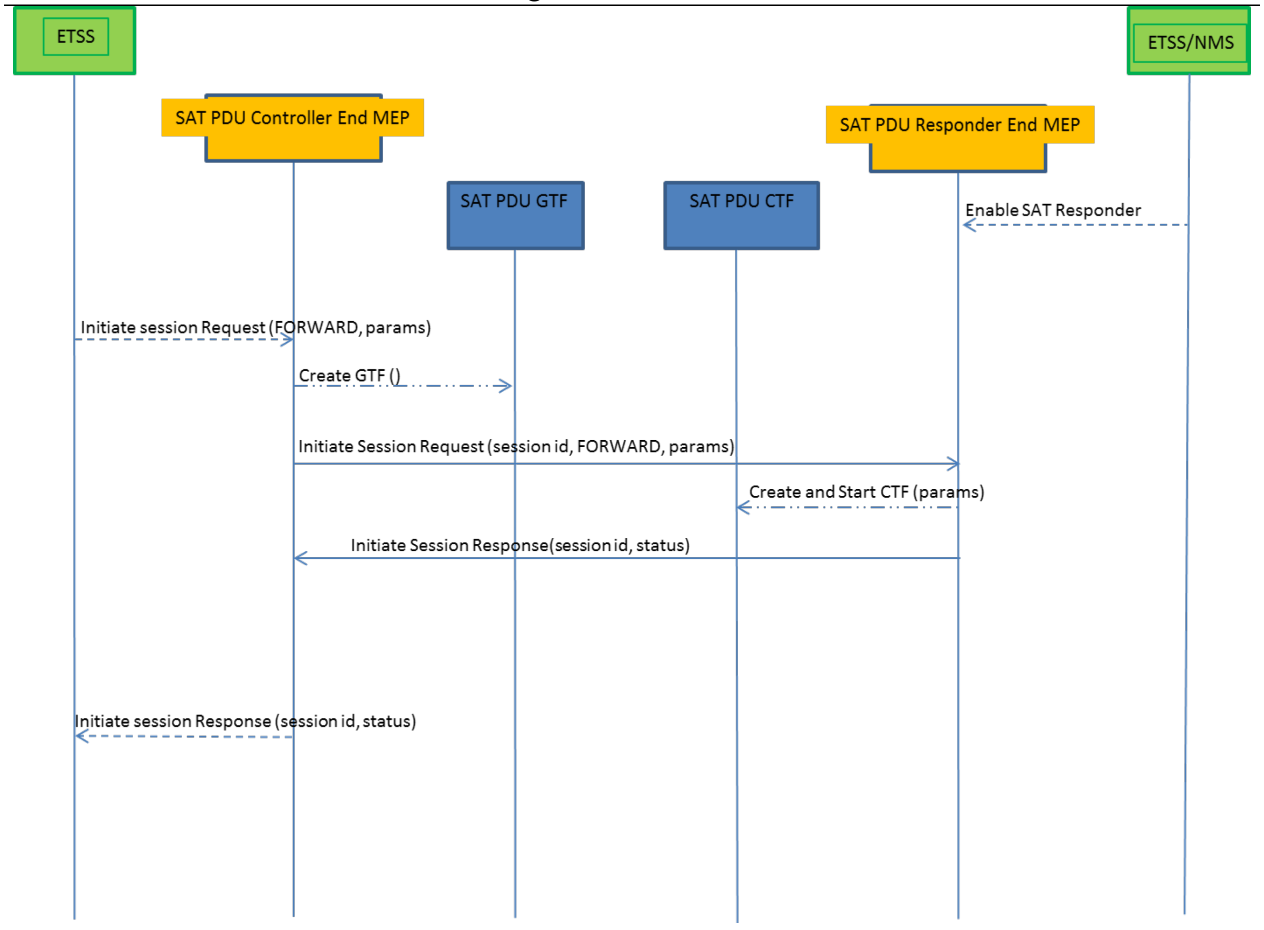

ETSS
ETSS/NMS
SAT PDU Controller End MEP
SAT PDU Responder End MEP
SAT PDU GTF
SAT PDU CTF
Enable SAT Responder
Initiate session Request (FORWARD, params)
Create GTF ()
Initiate Session Request (session id, FORWARD, params)
Create and Start CTF (params)
Initiate Session Response(session id, status)
Initiate session Response (session id, status)

Figure 39 Initiating a Session for a Forward Test

12.2.1.2 Running a Test

Figure 40 below depicts the session message interaction once the session is initiated.

The ETSS sends an external message to the Controller End MEP to start the test session. The Controller End MEP, in turn, uses an internal message to request the GTF start the test according to the parameters received in the original request from the ETSS. The GTF starts generating the test frames. It is assumed that the CTF, as part of the session initiation, was set to “listen” to test frames and count them.

When all the frames have been sent, the GTF informs the Controller End MEP using an internal message. The Controller End MEP sends a Stop Session Request Message to the Responder End MEP. The Responder End MEP uses an internal message to tell the CTF to stop counting. The Responder End MEP sends a Stop Session Request Response to the Controller End MEP. The Controller End MEP then requests the test session results from the Responder End MEP using a Fetch Session Results Message. The Responder End MEP uses an internal message to retrieve the results from its CTF and returns the results to the Controller End MEP using a Fetch Session Response Message.

Upon receipt of the test results, the Controller End MEP sends a Delete Session Request Message to the Responder End MEP. The Responder End MEP deletes the CTF using an internal message and sends a Delete Session Response Message to the Controller End MEP. The Controller End MEP deletes the GTF.

The Controller End MEP sends the test results to the ETSS using an external message. This could be done either in response to a request from the ETSS (as shown in the figure), or by the Controller End MEP pro-actively pushing the results to the ETSS after the session is complete.

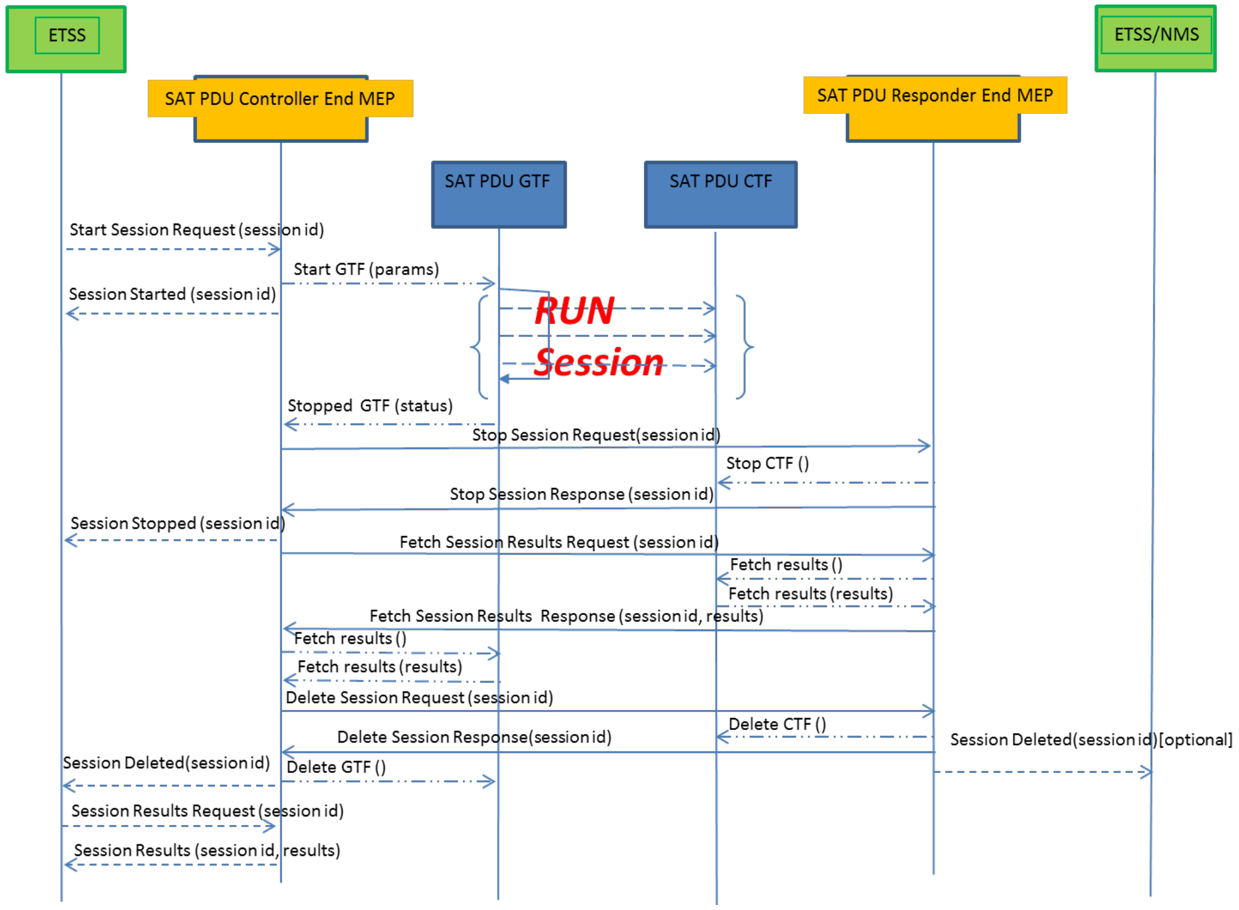

ETSS
ETSS/NMS
SAT PDU Controller End MEP
SAT PDU Responder End MEP
SAT PDU GTF
SAT PDU CTF
Start Session Request (session id)
Start GTF (params)
Session Started (session id)
RUN
Session
Stopped GTF (status)
Stop Session Request(session id)
Stop CTF ()
Stop Session Response (session id)
Session Stopped (session id)
Fetch Session Results Request (session id)
Fetch results ()
Fetch results (results)
Fetch Session Results Response (session id, results)
Fetch results ()
Fetch results (results)
Delete Session Request (session id)
Delete CTF ()
Delete Session Response(session id)
Session Deleted(session id)[optional]
Session Deleted(session id)
Delete GTF ()
Session Results Request (session id)
Session Results (session id, results)

Figure 40 Running a Test Session for a Forward Test

12.2.1.3 Abort Session

The previous section illustrates the mainline flow where the test completes successfully. Figure 41 below depicts the message interaction for a different flow, where the session is aborted.

If the ETSS needs to abort the session in the middle of the test session (i.e., before the GTF has finished sending all frames for the test), it sends an external message to the Controller End MEP requesting the test session be aborted.

The Controller End MEP, in turn, uses an internal message to request the GTF to abort the test and sends an Abort Session Request Message to the Responder End MEP. The Responder End MEP uses an internal message to request the CTF to abort the session and then deletes the CTF, and, optionally, sends an external message to notify the ETSS/NMS of the action.

Upon receipt of the Abort Session Response Message from the Responder End MEP, the Controller End MEP deletes the session. That is, the GTF is deleted.

The Controller End MEP uses an external message to inform the ETSS that the session has been aborted.

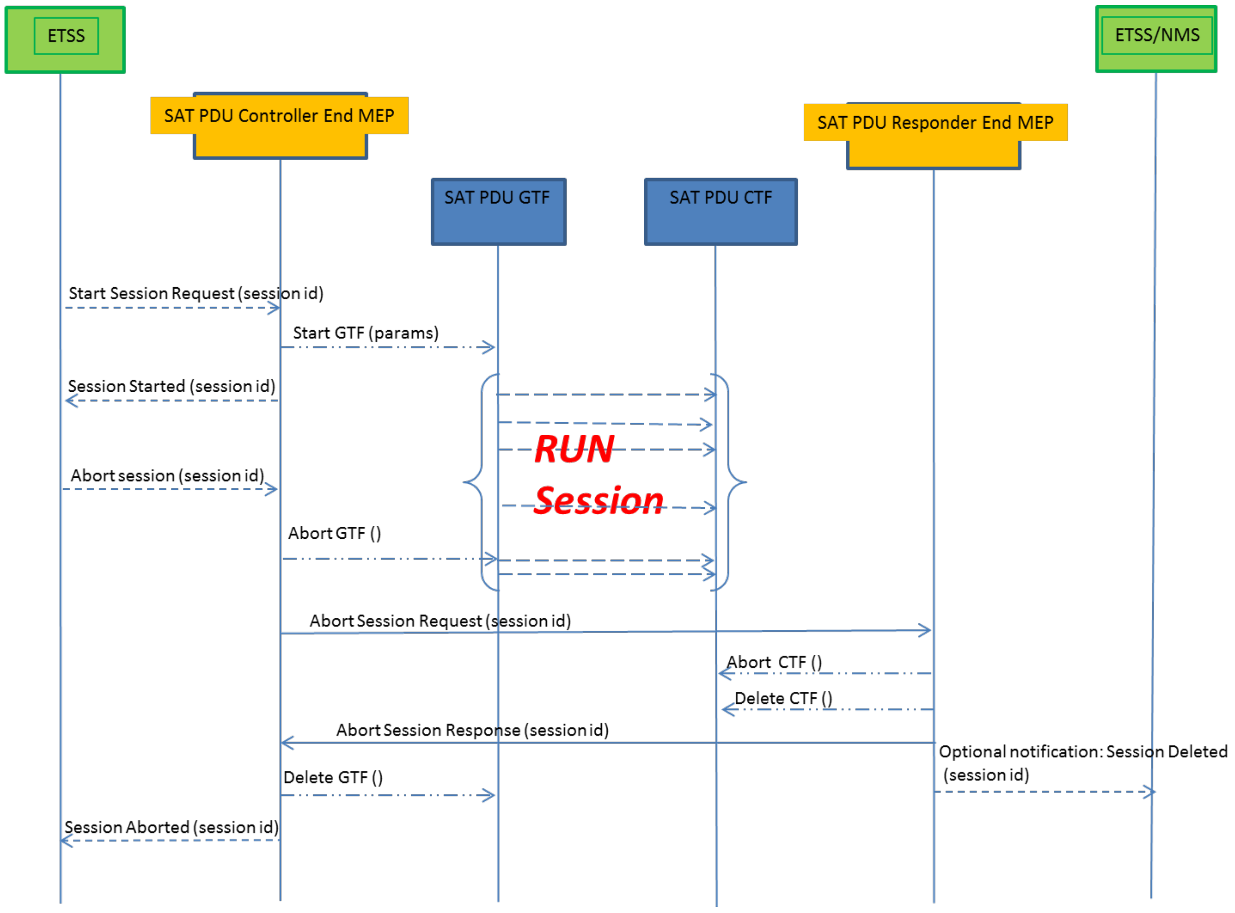

ETSS
ETSS/NMS
SAT PDU Controller End MEP
SAT PDU Responder End MEP
SAT PDU GTF
SAT PDU CTF
Start Session Request (session id)
Start GTF (params)
Session Started (session id)
RUN
Session
Abort session (session id)
Abort GTF ()
Abort Session Request (session id)
Abort CTF ()
Delete CTF ()
Abort Session Response (session id)
Optional notification: Session Deleted (session id)
Delete GTF ()
Session Aborted (session id)

Figure 41 Aborting a Session for a Forward Test

12.2.2 Backward Test Session

12.2.2.1 Session Initiation

Figure 42 below depicts the session initiation message interaction for a Backward test.

The ETSS sends an external message request to the Controller End MEP, requesting the initiation of a Backward test session. The Controller End MEP, in turn, uses an internal message to create a CTF and sends an Initiate Session Request Message to the Responder End MEP. The Responder End MEP uses an internal message to create a GTF, responds to the Controller End MEP with an Initiate Session Response Message and can notify the ETSS/NMS using an external message of its action. The Controller End MEP informs the ETSS of the results of its request using an external message. If successful, the result (in the form of Initiate Session Response) includes a Session ID, generated by the Controller End MEP.

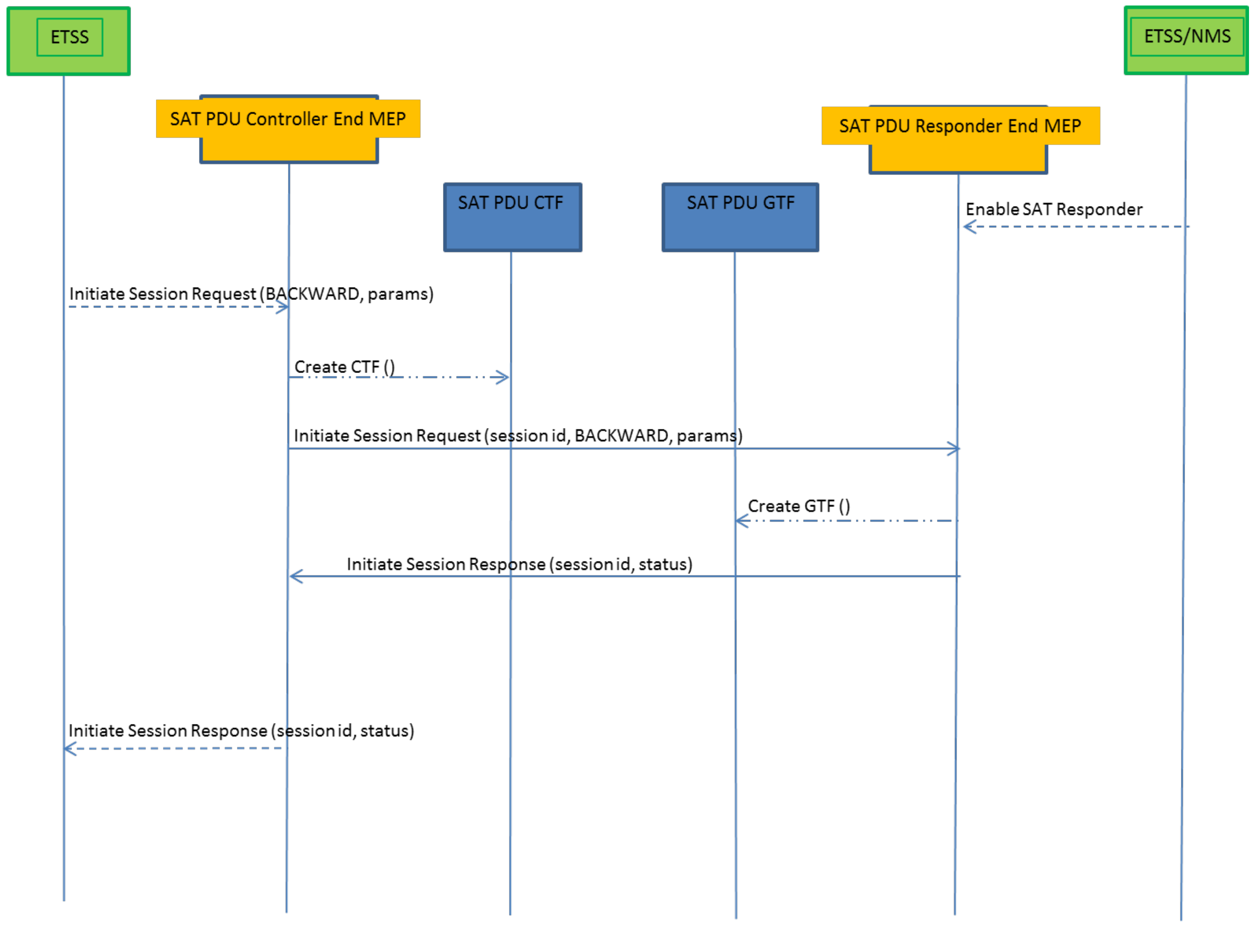

ETSS
ETSS/NMS
SAT PDU Controller End MEP
SAT PDU Responder End MEP
SAT PDU CTF
SAT PDU GTF
Enable SAT Responder
Initiate Session Request (BACKWARD, params)
Create CTF ()
Initiate Session Request (session id, BACKWARD, params)
Create GTF ()
Initiate Session Response (session id, status)
Initiate Session Response (session id, status)

Figure 42 Initiating a Session for a Backward Test

12.2.2.2 Running a Session

Figure 43 below depicts the session message interaction once the session is initiated.

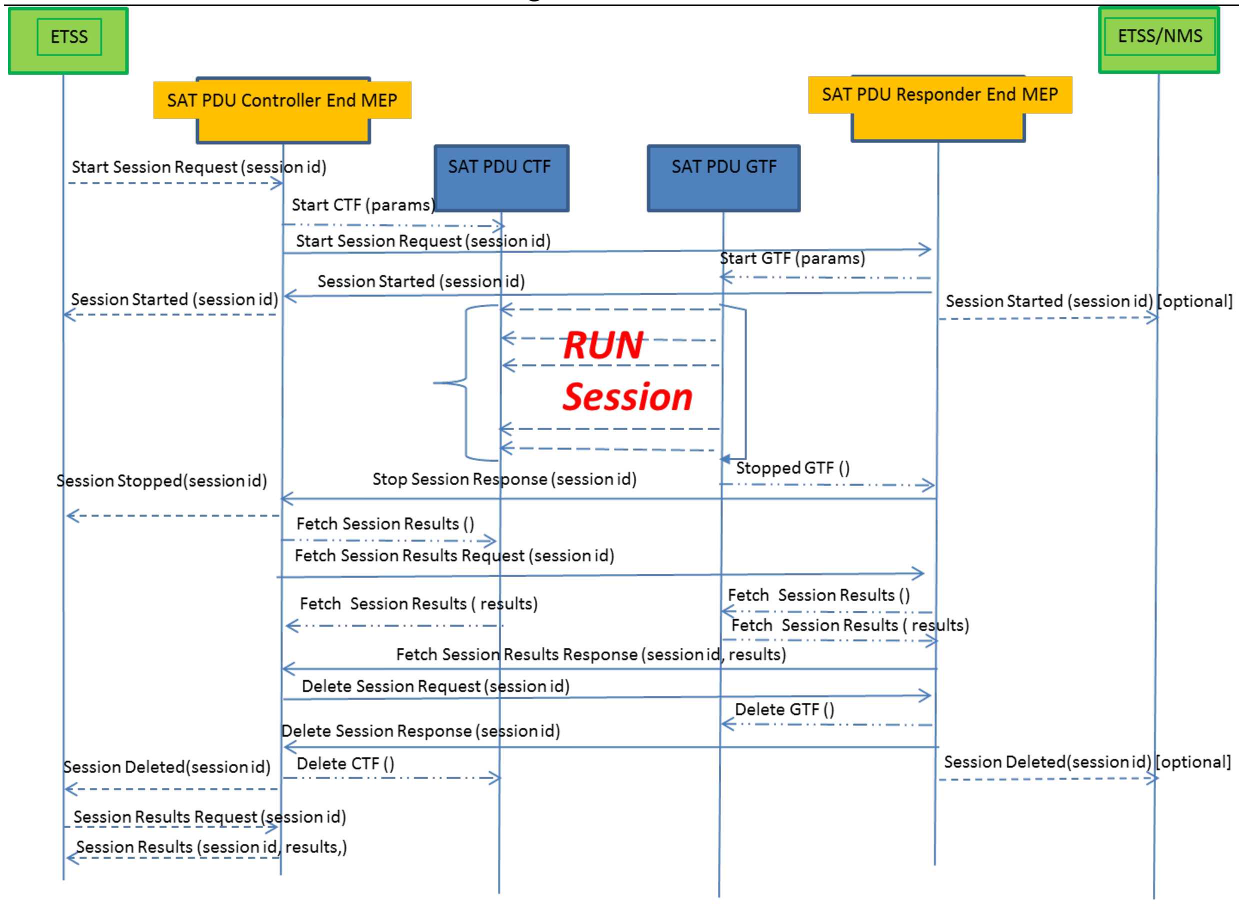

ETSS
ETSS/NMS
SAT PDU Controller End MEP
SAT PDU Responder End MEP
SAT PDU CTF
SAT PDU GTF
Start Session Request (session id)
Start CTF (params)
Start Session Request (session id)
Start GTF (params)
Session Started (session id)
Session Started (session id)
Session Started (session id) [optional]
RUN Session
Stopped GTF ()
Session Stopped(session id)
Stop Session Response (session id)
Fetch Session Results ()
Fetch Session Results Request (session id)
Fetch Session Results (results)
Fetch Session Results ()
Fetch Session Results (results)
Fetch Session Results Response (session id, results)
Delete Session Request (session id)
Delete GTF ()
Delete Session Response (session id)
Session Deleted(session id)
Delete CTF ()
Session Deleted(session id) [optional]
Session Results Request (session id)
Session Results (session id, results,)

Figure 43 Running a Session for a Backward Test

The ETSS sends an external message to the Controller End MEP to start the test session. The Controller End MEP, in turn, uses a Start Session Request Message to request the Responder End MEP start the test session according to the parameters received in the Initiate Session Request Message. The Responder End MEP uses an internal message to activate the GTF and returns a Start Session Response Message to the Controller End MEP. The GTF starts generating the test frames according to the session attributes. It is assumed that the CTF, as part of the session initiation, was set to “listen” to test frames and count them.

When all the frames have been sent, the GTF informs the Responder End MEP via an internal message. The Responder End MEP informs the Controller End MEP that all frames have been sent using a Stop Session Response Message. The Controller End MEP then requests the test results from the CTF using an internal message and from the Responder End MEP using a Fetch Session Results Request Message. The Responder End MEP uses an internal message to request the test session results from the GTF and returns the results from the GTF to the Controller End MEP using a Fetch Session Results Response Message.

Upon receipt of the test results the Controller End MEP sends a Delete Session Request Message to the Responder End MEP. The Responder End MEP deletes the GTF using an internal message and sends a Delete Session Response Message to the Controller End MEP. The Controller End MEP deletes the CTF.

The Controller End MEP then can either send (i.e., push) the test results to the ETSS or the ETSS requests (polls) the results from the Controller End MEP using external messages –the latter option is shown in the figure.

12.2.2.3 *Abort Session*

The previous section illustrates the mainline flow where the test completes successfully. Figure 44 below depicts message interaction for a different flow, where the session is aborted.

If the ETSS needs to abort the session in the middle of the test session (i.e., before the GTF has finished sending all the frames for the test), it sends an external message to the Controller End MEP requesting the test session be aborted.

The Controller End MEP, in turn, sends an Abort Session Request Message to the Responder End MEP. The Responder End MEP uses an internal message to request the GTF to abort the session and then deletes the GTF, and, optionally, sends an external message to notify the ETSS/NMS of the action.

Upon receipt of the Abort Session Response Message from the Responder End MEP, the Controller End MEP uses an internal message to delete the CTF. The CTF is not deleted until the Controller End MEP receives notification that the Responder End MEP has deleted the GTF to avoid test traffic from leaking out to the OVC under test.

The Controller End MEP uses an external message to inform the ETSS that the session has been aborted.

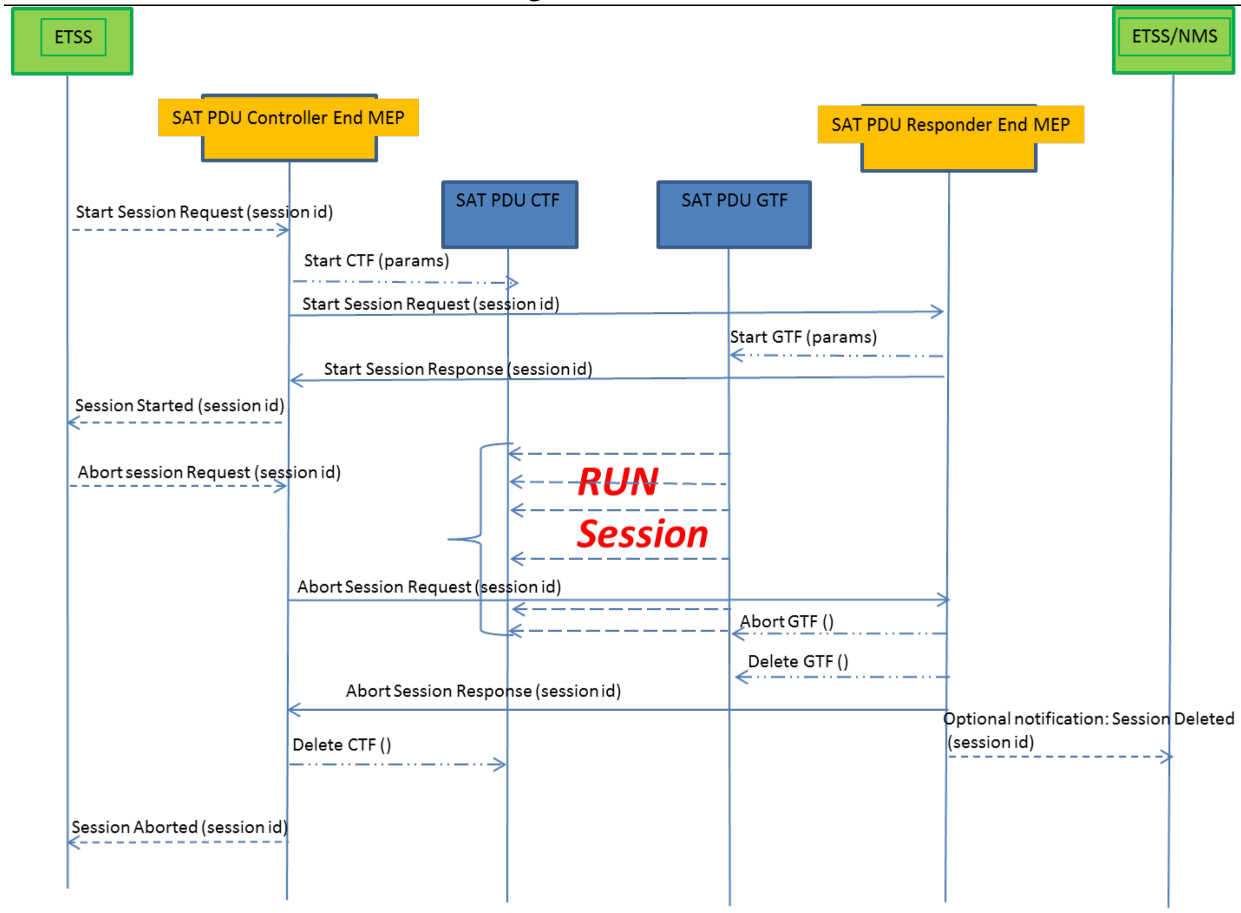

ETSS
ETSS/NMS
SAT PDU Controller End MEP
SAT PDU Responder End MEP
SAT PDU CTF
SAT PDU GTF
Start Session Request (session id)
Start CTF (params)
Start Session Request (session id)
Start GTF (params)
Start Session Response (session id)
Session Started (session id)
Abort session Request (session id)
RUN
Session
Abort Session Request (session id)
Abort GTF ()
Delete GTF ()
Abort Session Response (session id)
Optional notification: Session Deleted (session id)
Delete CTF ()
Session Aborted (session id)

Figure 44 Aborting a Test Session for a Backward Test