ASN.1 Requirements Wireless Environments (WAVE) -- Test Control Interface ASN1 Specification

Document Mnemonics:	WAVE-TCIS-ASN1-V3
Revision:	[V3.0.0]
Revision Date:	01/03/2021

Document Submission	Aaron Moore
Company	OmniAir Consortium
Contact Information	O: 1-202-815-6138
	amoore@omniair.org

Table of Contents

1 Revision History 5
2 Scope 5
3 References 5
3.1 Normative References 5
3.2 Informative References 6
4 Abbreviations 6
5 Test System 7
5.1 Architecture 7
5.2 Hardware equipment 7
5.2.1 Test System 8
5.3 DSRC radio 8
5.4 C-V2X radio 8
5.5 Interface Requirements 9
5.5.1 Test System Interface (TS ←→ SUT) 9
5.5.2 Interface to Radio (TS ←→ Radio) 9
5.5.3 Constraints 9
6 TCI Message Protocol 9
6.1 General 9
6.1.1 TS sends a request to SUT and receives a *Response* 10
6.1.2 SUT sends an unsolicited *Indication* to the TS 10
6.1.3 TS sends a request and receives information from the SUT 11
6.1.4 SUT sends an unsolicited Exception to the SUT 11
6.2 Transport Protocol 11
7 Test Control Interface Messages 12
7.1 Shared message structure 12
7.2 Test Control Interface Modules 13
8 Common TCI modules 13
8.1 TCI-wsm module 13
8.1.1 Request messages 14
8.1.2 Content Type 20
8.1.3 Signer Identifier Type 20
8.2 TCI-ip module 21
8.2.1 Base Classes 21
8.2.2 Request messages 21
8.3 Response, ResponseInfo, Indication and Exception messages 25
8.3.1 Response messages 25

8.3.2 Indication messages 25
8.3.3 ResponseInfo messages 26
8.3.4 Exception messages 27
9 TCI frames 27
9.1 TCI80211 frame 27
9.1.1 Supported use cases 27
9.1.2 *Request* Messages 28
9.1.3 *Response* messages 29
9.1.4 *Indication* messages 29
9.1.5 *Exception* messages 29
9.2 TCI16094 frame 30
9.2.1 Supported use cases 30
9.2.2 *Request* Messages 31
9.2.3 *Response* messages 33
9.2.4 *Indication* messages 33
9.2.5 *ResponseInfo* messages 33
9.2.6 *Exception* messages 33
9.3 TCI16093DSRC frame 33
9.3.1 Supported use cases 33
9.3.2 *Request* messages 35
9.3.3 *Response* messages 37
9.3.4 *Indication* messages 37
9.3.5 *ResponseInfo* messages 38
9.3.6 *Exception* messages 38
9.4 TCI16093PC5 frame 38
9.4.1 Supported use cases 38
9.4.2 *Request* messages 40
9.3.3 *Response* messages 43
9.3.4 *Indication* messages 43
9.3.5 *ResponseInfo* messages 44
9.3.6 *Exception* messages 44
9.4 TCI29451 frame 45
9.4.1 Request messages 45
9.4.2 *Response* messages 49
9.4.3 *Indication* messages 49
9.4.4 *ResponseInfo* messages 50
9.4.5 *Exception* messages 50
9.5 TCI31611 frame 50
9.5.1 Supported use cases 50

9.5.2 *Request* messages 51
9.4.2 *Response* messages 54
9.4.3 *Indication* messages 54
9.4.4 *ResponseInfo* messages 54
9.4.5 *Exception* messages 55
9.6 TCISutControl frame 55
9.6.1 Supported use cases 55
9.6.2 *Request* messages 56
9.6.3 Response messages 58
9.6.4 *ResponseInfo* messages 58
9.6.5 *Exception* messages 59
Appendix A: TCI protocol ASN.1 definition 59
Appendix B: TCIProxyCv2X 63
General: 63
Proxy Setup: 63
Supported use cases 63
Request messages 64
Open Issues 65

1 Revision History

V2.0.0	February 24, 2020	Initial Draft of Version 2 – Aaron Moore
V3.0.0	January 03, 2021	Initial Draft of Version 3 – Aaron Moore

2 Scope

This document provides the message interface and protocol to be used between a Test System (TS) and a System Under Test (SUT). The protocol is defined using ASN.1 and referenced in Appendix A.

The intent of this document is to provide an overview of the protocol. It explains the architecture of the protocol, main use cases and how the messages are structured. Details of the type definitions are not described in this document. Instead, the reader is required to review the ASN.1 definition.

3 References

3.1 Normative References

The following referenced documents are necessary for the application of the present document.

[1] WAVE802.11-TSS&TP (V1.2.0): "Conformance test specifications for Wireless Access in Vehicular Environments (WAVE) — 802.11 Test Suite Structure and Test Purposes (TSS & TP)". Revision date: 10/09/2017

[2] WAVEMCO-TSS&TP (V1.3.0): "Conformance test specifications for Wireless Access in Vehicular Environments (WAVE) — Multi-channel Operation Test Suite Structure and Test Purposes (TSS & TP)". Revision date: 10/08/2017

[3] WAVENS-TSS&TP (V1.3.3): "Conformance test specifications for Wireless Access in Vehicular Environments (WAVE) — Networking Services Test Suite Structure and Test Purposes (TSS & TP)". Revision date: 10/08/2017

[4] WAVE-16092-TSS&TP (V0.6.0): "Conformance test specifications for Wireless Access in Vehicular Environments (WAVE) — Security Services Test Suite Structure and Test Purposes (TSS & TP)". Revision date: 10/08/2017

[5] J2945/1-TSS&TP (V0.5.5): "Conformance test specifications for SAE J2945/1 - On-board System Requirements for V2V Safety Communications Test Suite Structure and Test Purposes (TSS & TP)". Revision date: 10/08/2017

[6] "DSRC Proxy", (V0.5.0), Revision date: 11/6/2015.

[7] IEEE Std. 802.11™-2012: "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications".

[8] IEEE Std 1609.3-2016 "IEEE Standard for Wireless Access in Vehicular Environments (WAVE) — Network Services".

[9] SAE J2945/1 (J2945/1_201603): "On-Board System Requirements for V2V Safety Communications".

[10] SAE J2735 (2016-01): "Dedicated Short Range Communication (DSRC) Message Set Dictionary".

[11] IEEE Std 1609.2-2016 “IEEE Standard for Wireless Access in Vehicular Environments (WAVE) — Security Services”.

[12] IEEE Std. 1609.4-2016 “IEEE Standard for Wireless Access in Vehicular Environments (WAVE) -- Multi-Channel Operation”.

[13] 3GPP 36.521-1-2019 (E-UTRA) UE Conformance Specification; Radio transmission and reception; Part 1: Conformance testing.

[14] 3GPP 36.521-3-2019 (E-UTRA) UE Conformance Specification; Radio transmission and reception; Part 3: Radio Resource Management (RRM) Conformance Testing

[15] 3GPP 36.523-1-2019 (E-UTRA) and (OPC); UE conformance specification; Part 1: Protocol conformance specification

[16] SAE J3161/1-2020 “On-Board System requirements for LTE V2X V2V Safety Communications

3.2 Informative References

The following referenced documents are not necessary for the application of the present document, but they assist the user regarding a particular subject area.

[i.1] ETSI EG 202 798 (V1.1.1): "Intelligent Transport Systems (ITS); Testing; Framework for conformance and interoperability testing".

[i.2] ETSI ETS 300 406 (1995): "Methods for testing and Specification (MTS); Protocol and profile conformance testing specifications; Standardization methodology".

4 Abbreviations

For the purposes of the present document, the following abbreviations apply:

ABS	Anti-lock Braking System
ASN	Abstract Syntax Notation
BSM	Basic Safety Message
CH	Channel
CPU	Central Processing Unit
DSRC	Dedicated Short Range Communications
GPS	Global Positioning System
ICMP	Internet Control Message Protocol
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
ISO	International Organization for Standardization
ITS	Intelligent Transport Systems
IUT	Implementation Under Test
NTP	Network Time Protocol
OER	Octet Encoding Rules
PC	Personal Computer
PDU	Protocol Data Unit
PSID	Provider Service Identifier
RCPI	Received Channel Power Indicator

RX	Receive
SAE	Society of Automotive Engineers
SUT	System Under Test
TCI	Test Control Interface
TCIA	Test Control Interface Application
TCP	Transport Control Protocol
TP	Test Purposes
TRI	Tester Radio Interface
TS	Test System
TX	Transmit
UC	Use Case
UDP	User Datagram Protocol
UPER	Unaligned Packed Encoding Rules
WAVE	Wireless Access in Vehicular Environments
WME	WAVE Management Entity
WSA	WAVE Service Advertisement
WSM	WAVE Short Message

5 Test System

5.1 Architecture

The Test System used to support tests listed in [1], [2], [3], [4], and [5] is described in Figure 1. The test system is designed to simulate valid and invalid protocol behaviors and analyze the reaction of the IUT.

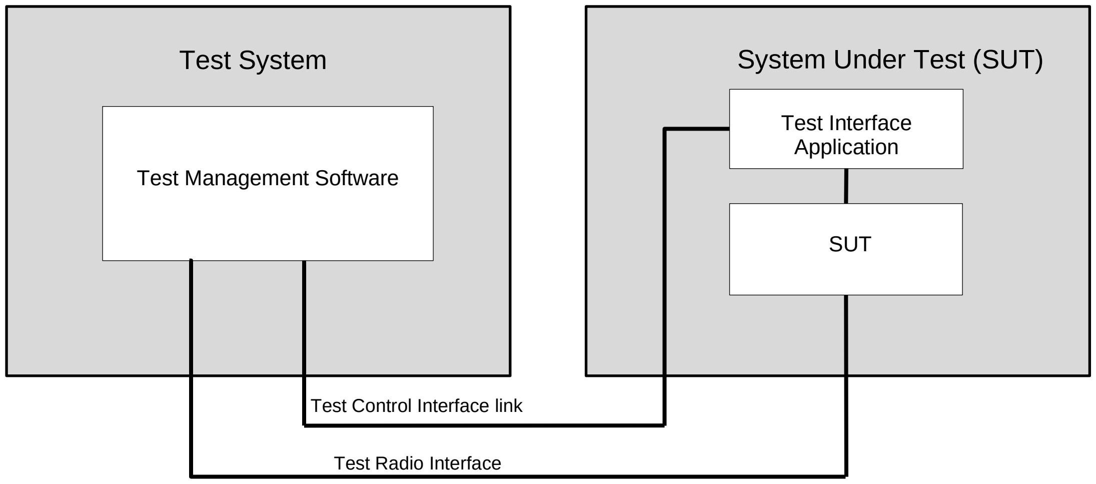

Figure 1: General Architecture

5.2 Hardware equipment

The system is implemented according to Figure 2. The test system is comprised of Test Management Software running on a PC (or laptop). The PC is physically connected to the SUT via an Ethernet cable supporting an IPbased connection to transfer control and test data to and from the SUT. This connection corresponds to the Test Control Interface as depicted on Figure 1. The Wired Ethernet connection may be substituted by a wired USB cable if the device supports IPv4-based data exchanges (e.g., support of RNDIS protocol) or a wireless Ethernet connection if the SUT does not support a wired connection.

The Test System connects to an external radio using a separate wired Ethernet link. Theradio is used to transfer wireless data messages between the Test System and the SUT. This interface is depicted as the Test Radio Interface on Figure 1.

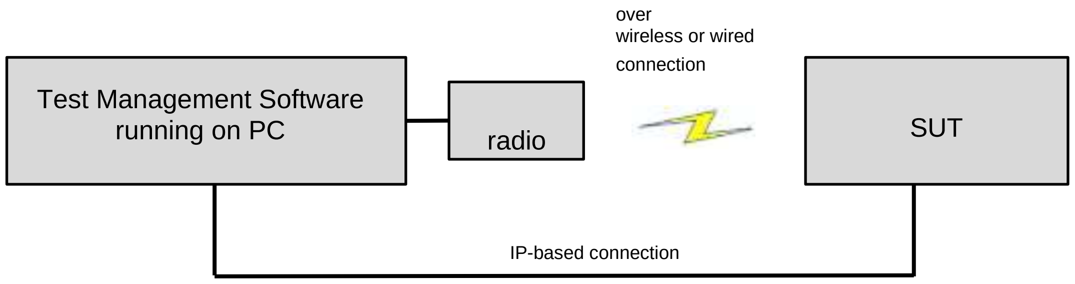

Figure 2: Test System Implementation

5.2.1 Test System

The main hardware component of the Test System is a standard PC. Its role is to host the execution of the Test Management Software, manage the test flow and generate test reports. To construct a Test System, the following points must be considered:

- No firewall interference with traffic generated by the Test System and/or SUT.
- Use of a synchronized time reference for the SUT and the test system. The Test System may be synchronized to UTC via a Network Time Protocol (NTP), whereas the SUT may use GPS for time synchronization and be adjusted to UTC via data post processing.
- The Test System processes must be granted unrestricted control to the telecommunication hardware.

Time synchronization between the Test System and the SUT must be checked before starting any test session, as it can be the source of unpredictable SUT behavior and generate incoherent results. For example, most protocol messages feature a time tag used by the receiver to determine if the information it carries is still valid; if the test system is not synchronized, all messages it sends will be considered either as coming from the future or past and be discarded.

The Test System must be equipped with at least one network interface supporting IPv4 protocol link independent of DSRC protocol link to exchange control and test data messages with the SUT.

TCI message exchanges are established using UPD over IPv4-based protocols. Any references to the IPv6 protocol are used regarding the wireless exchanges since the IPv4 protocol is not supported for over-the-air transmissions.

5.3 DSRC radio

To monitor and test DSRC message exchanges, a DSRC radio that fully supports the IEEE 802.11 standard [7] is included in the Test System. The DSRC radio acts as a bridge and passes all messages to and from the Test System which performs message encoding/decoding and verification. The interface between Test System and DSRC radio is covered in a separate document [6].

5.4 C-V2X radio

To monitor and test C-V2X message exchanges, a C-V2X radio that fully supports the 3GPP 36.521-1 [13], 36.521-3 [14], and 36.523-1 [15] standards. The C-V2X radio acts as a bridge and passes all messages to and from the Test System which performs message encoding/decoding and verification. The interface between Test System and DSRC radio is covered in Appendix B of this document.

5.5 Interface Requirements

5.5.1 Test System Interface (TS ←→ SUT)

This clause lists requirements for the Test System Interface between the Test System and the Test Control Interface Application (TCIA) running on the SUT:

- The Test System shall communicate with the SUT using the commands described in this document.
- All commands shall be issued using UDP messages. Commands can be used to change the SUT state, operating mode, configure data on the SUT, stimulate the SUT, and observe how the SUT responds to external stimulations.
- The Test System shall send UDP messages to the SUT using IPv4 protocol. The SUT will run the TCIA. This application will decode commands received via UDP messages and use the appropriate software interface to execute the command.
- The TCIA shall listen for the command coming from the Test System using the UDP port (**13001**).
- The TCIA shall send the responses to the Test System UDP port from which the initial *SetInitialState* request came from.

5.5.2 Interface to Radio (TS ←→ Radio)

This clause lists requirements for the interface between the TS and the radio.

- The SUT communicates to the specific radio using wireless protocol for that specific radio medium.
- The radio translates the received WSM messages and sends them to the TS using UDP protocol.
- The radio receives UDP packets from the TS and transmits them as WSM over the medium specific protocol.
- The conversion between the WSM and UDP protocol is performed as described in Appendix [X] of this document.

5.5.3 Constraints

This document only describes the interface between the Test System and the TCI Application. Implementation details of the TCI Application or the SUT is outside the scope of this document.

6 TCI Message Protocol

6.1 General

This document primarily focuses on the Test Control Interface as depicted on Figure 1. The communication between the Test System and the SUT is achieved using messages flowing using a UDP protocol.

The message exchange format is laid out as follows

- **Request**: This message is initiated from the Test System to the SUT to stimulate the SUT to trigger requested functionality.
- **Response**: This message is sent from the SUT to the Test System indicating an acceptance of the *Request* by the SUT. Acceptance means ability of the SUT to decode and interpret the message to initiate a sequence of changes at the SUT.
- **ResponseInfo**: This message is sent from SUT to the Test System and contains parameter information requested by SUT, for example retrieval of SUT default settings.
- **Indication**: An event message is sent from the SUT to the Test System indicating the SUT has received a DSRC message or an SUT event occurred.

- **Exception**: This message is sent from the SUT to the Test System. This message is used to report all exception conditions (i.e., INFO/WARNING/ERROR) generated in the SUT to the Test System. Depending on the exception severity, the TS may initiate recovery (i.e., reset to the initial state), or continue its operation.

The TS expects to receive *Response* or *Exception* messages within **50ms** after the SUT received a *Request* message. If no *Response* or *Exception* is received, the TS will attempt to re-initialize the SUT or may require user assistance.

The typical message exchanges are described below:

6.1.1 TS sends a request to SUT and receives a *Response*

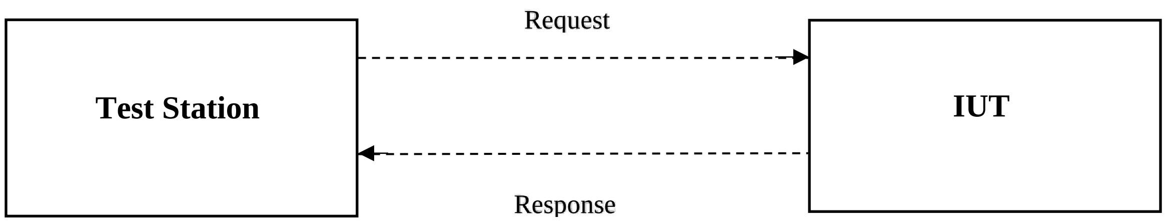

Figure 3

The communication exchange is initiated by the TS. The TS sends a *Request*. The SUT responds with a *Response* containing a result code indicating success of an operation or an exception. In the latter case, the *Response* message includes information about the exception.

The response is an acknowledgement that the SUT received the test system's request and will be acting on it. It then executes the request. It is the TS that determines if the test passes or fails based on the result of the test.

6.1.2 SUT sends an unsolicited *Indication* to the TS

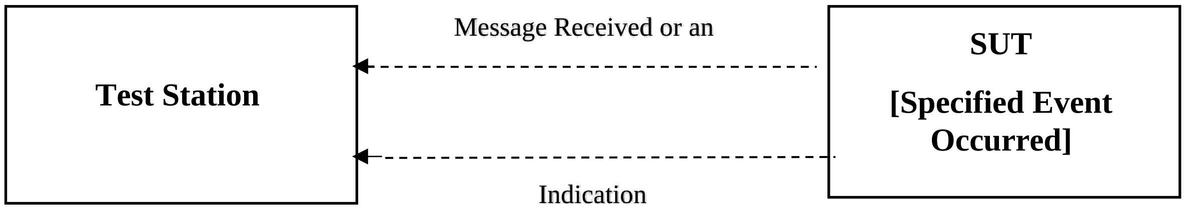

Figure 4

This communication exchange is initiated by the SUT. The SUT may send an unsolicited indication to the TS each time a packet is received and processed by the SUT or an event occurred on the SUT. Normally, the SUT will start (or stop) sending *Indications* after it is triggered by the TS. The TS never replies to such messages.

6.1.3 TS sends a request and receives information from the SUT

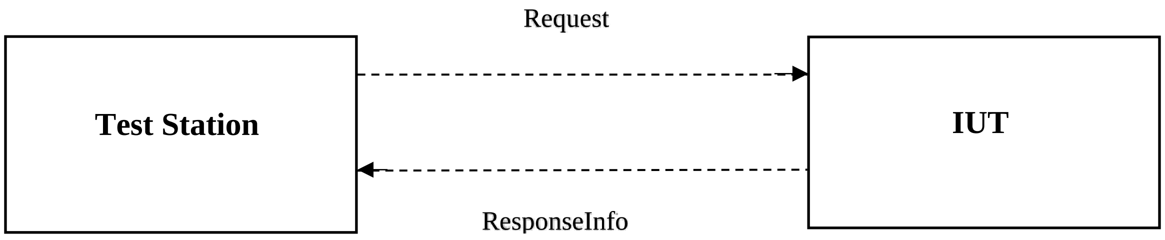

Figure 5

The TS needs to obtain information from the SUT, e.g., the IPv6 address of the wireless interface. The TS sends a request message. The SUT does not sends a *Response*, but instead replies with a *ResponseInfo* containing the requested information.

6.1.4 SUT sends an unsolicited Exception to the SUT

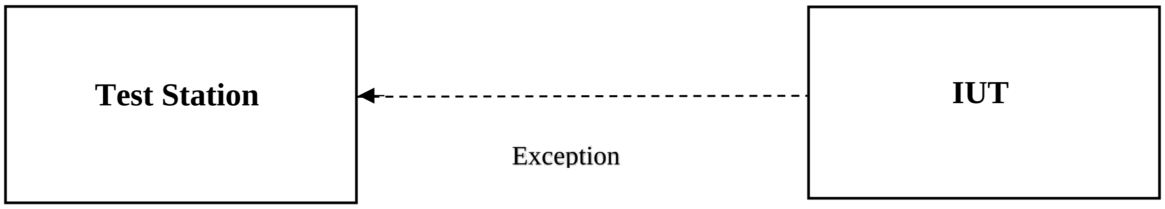

Figure 6

The SUT needs to inform the TS about an exception. The SUT sends an *Exception* message to the TS. TS does not reply to the SUT. This *Exception* message may be generated at any time and does not require a *Request* from the TS.

Message specification is defined using ASN.1. It is provided in the Appendix A. The default encoding for all TCI messages is using **OER** encoding. Additional TCI message encodings, e.g., UPER, may also be supported. Note, that some TCI messages may contain a parameter containing a message payload. The content of the payload must be encoded to be directly transferrable to the target message payload without re-encoding.

A log of all the message exchanges with the system defined timestamps are maintained in a log file on the Test System; this helps in correlating if the test result is not as expected.

6.2 Transport Protocol

The communication between the TS and SUT uses UDP protocol messages flowing via IPv4-based link. The IP addresses for TS and SUT can be selected from the following ranges:

Testing System:	192.168.23.1 … 192.168.23.127, subnet 255.255.255.0
SUT:	192.168.23.128 … 192.168.23.254, subnet 255.255.255.0

To initiate the connection, the TS sends the initial *Request* message to a pre-defined UDP destination port (*defaultTCIAPort = 13001*), which the SUT opens to listen for incoming messages. When the SUT receives the first *Request* message from the TS, it saves the UDP source port of this request as *defaultTSPort.* The SUT uses the *defaultTSPort* UDP port to send *Response*, *ResponseInfo* messages as well as unsolicited *Indication* and *Exception* messages.

The TS must keep the *defaultTSPort* unchanged during continuous testing sequence until the TS and/or the SUT is reset, test sequence is interrupted, or another similar event takes place. When the testing is resumed, the previously described process is repeated: the SUT waits for the initial *Request* message on the *defaultTCIAPort*, stores the source port as the *defaultTSPort* and sends the response back to the *defaultTCIAPort*.

The SUT can receive the initial *Request* message of type *SetInitialState*, or *RequestSutAvailability*. The latter case will apply when the SUT is recovering from the previously received requests for *Shutdown* or *Restart*. The TS may also start the test execution with the *SetTestId*. Regardless which message is SUT receives first from the TS, the SUT will use the UDP source port from the first *Request* message from the TS and use it as the destination UDP port when the SUT sends messages to the TS.

Table 1 TS and SUT default UDP ports configuration

Parameter	Description	Value
defaultTCIAPort	UDP port used by the TCIA to receive request from TS.	13001
defaultTSPort	UDP port used by TS to listen for SUT indications and responses.	The source UDP port used by the TS for sending the *SetInitialState* or *RequestSutAvailability* request messages.

7 Test Control Interface Messages

7.1 Shared message structure

All messages defined in this specification are grouped under the common root type called *TCIMsg*, which contains the following parameters:

Parameters	Definition	Description
version	Integer (0..255)	For this revision of specification, version shall be set to 2.
timestamp	Time64	Timestamp provided by the message sender. Timestamp measures the the difference in milliseconds, between the current time and midnight, **January 1, 1970 UTC**
frame	CHOICE{ TCI16093DSRC, TCI16093PC5, TCI16094, TCI29451, TCI31611, TCI80211}	Current TCI frames defined in this specification.

Messages for all frames have the same defined structure. The following examples describes TCI16093Event for communications.

```
TCI16093DSRC ::= CHOICE{
request
response
indication
responseInfo
exception
…
}
```

```
TCI16093PC5 ::= CHOICE{
request
response
indication
responseInfo
exception
…
}
```

The following sections provide the top-level definition of the TCI frame. Appendix A: provides message and type definitions in ASN.1 format.

7.2 Test Control Interface Modules

TCI protocol is defined in the modules listed in the Table 2.

Table 2 TCI protocol modules

ASN.1 Module	Description
TCI-16093DSRC	Frame and message definition used for testing 1609.3 DSRC
TCI-16093PC5	Frame and message definition used for testing 1609.3 PC5
TCI-16094	Frame and message definition used for testing 1609.4
TCI-29451	Frame and message definition used for testing 2945/1
TCI-31611	Frame and message definition used for testing 3161/1
TCI-80211	Frame and message definition used for testing 802.11
TCI-CommonTypes	Common types shared across TCI modules
TCI-Dispatcher	Root module aggregating all other frame specific messages
TCI-EventHandling	Common event-handling types shared by other modules
TCI-SutControl	Device-level commands for controlling SUT
TCI-indication	Common indication messages shared by other modules
TCI-ip	Request messages for sending and receiving IPv6 packets
TCI-responseInfo	Returns Version Information from IUT
TCI-wsm	Request messages for sending and receiving WSM packets
TCIproxyCv2x	Optional for implementation, frame, and definition for use of proxy

For example, several TCI frames trigger transmission of WSM. Those requests are defined in the *TCI-wsm* module and included into the corresponding modules *TCI-16093DSRC*, *TCI-16093PC5*, *TCI-80211*, etc. by reference. Similarly, requests to transmit IPv6 packets are defined in the *TCI-ip* module and imported into the modules *TCI-16093DSRC*, *TCI-16093PC5*, *TCI-16094*, etc. by reference.

8 Common TCI modules

This section describes common messages shared by TCI frames.

8.1 TCI-wsm module

The *TCI-wsm* module defines request messages from the TS to the SUT to trigger transmission and/or reception of WSMs. It also includes messages for management of the corresponding parameters and service tables on the SUT.

Many WSM parameters including *PSID*, *MACaddress*, *RadioInterface*, *RepeatRate*, etc., are defined by reusing the corresponding types from IEEE 1609.3 [8]. This specification adopts definitions of these parameters from the standard [8]. For the ASN.1, TCI imports these data types from the corresponding definitions of the standard.

Conventions for time and geo-location data representation are adopted from the SAE J2735 [10].

IEEE 1609.3 uses *UPER* encoding while TCI specification uses, by default, OER encoding. Due to encoding difference, the same parameters values may have different representation once encoded for transmission as WSM compared to TCI messages.

8.1.1 Request messages

8.1.1.1 SetInitialState

This request is used to set the SUT in initial condition. The initial condition defines the initial state in which the SUT must be to carry out each test case. This message also must clear information from the following MIB tables *ProviderServiceRequestTable*, *UserServiceRequestTable*, as defined in IEEE1609.3 [8].

8.1.1.2 SetWsmTxInfo

This request is used to configure device parameters before transmitting WSMs.

```
SetWsmTxInfo ::= SEQUENCE{
    psid                      Psid,
    radio                     RadioInterface,
    security                  SecurityContext,
    transmitPowerLevel        TXpower80211 OPTIONAL
    infoElementsIncluded      WaveElementsIncluded DEFAULT ‘000000000000000000000000'B
    userPriority              UserPriority OPTIONAL,
    channelIdentifier         ChannelNumber80211 OPTIONAL,
    dataRate                  DataRate OPITONAL,
    timeslot                  TimeSlot OPTIONAL,
    repeatRate                RepeatRate OPTIONAL,
    destinationMACAddr        MACaddress OPTIONAL,
    channelLoad               Opaque OPTIONAL,
    expiryTime                Time64 OPTIONAL,
    Payload                   Opaque OPTIONAL,
    …,
    txPoolId                    TxPoolId OPTIONAL
    flowID                    FlowIdentifier OPTIONAL
    }
```

Table 3 SetWsmTxInfo parameters

Parameters	Explanation
psid	Provider Service identifier as defined in 1609.3 [8].
radio	The structure contains radio device (radio0, radio1, etc.) and antenna port for transmission of WSMs
security	The structure security context including content type of payload (i.e. BSM, WSA) for selecting appropriate security profile; security type (i.e. unsecure, signed, etc.); optional reference to a certificate hashID.
transmitPowerLevel	Transmit power level as defined in 1609.3 [8].
infoElementsIncluded	A bit flag indicating which optional WAVE Info Elements included in the WSM-N-Header
userPriority	User priority as defined in 1609.3 [8].
channelIdentifier	Channel number as defined in 1609.3 [8].
dataRate	Data rate as defined in 1609.3 [8].
timeslot	Time slot or continuous channel usage as defined in 1609.3 [8].
repeatRate	Repeat rate for messages as defined in 1609.3 [8] as number of messages per 5 sec intervals. Additionally, it can be set to 0 for transmitting a single message.
destinationMACAddr	Destination MAC address for the destination as defined in 1609.3 [8]. Default value set for broadcast transmissions.

channelLoad	Channel load as defined in 1609.3 [8].
expiryTime	Expiry time as defined in 1609.3 [8]. This is an optional parameter.
Payload	WSM message payload excluding message length field.
txPoolId	Transmit Pool Identifier
flowID	Defines flow parameters via SetFlowConfig

Specific details for each type definition are listed in the ASN.1 specification referenced in Appendix A.

8.1.1.3 StartWsmTx

This request is used to initiate transmission of WSMs by the SUT. Information from this request can be used to invoke *WSM-WaveShortMessage.request* from 1609.3 [8] and must be preceded by StartWsmTxInfo to set up the transmitter parameters.

```
StartWsmTx ::= SEQUENCE {
 psid                 Psid OPTIONAL,
 radio                RadioInterface,
 repeatRate           RepeatRate OPTIONAL,
 payload              Opaque OPTIONAL,
 ...
}
```

Table 4 StartWsmTx parameters

Parameters	**Explanation**
psid	Provider Service identifier as defined in 1609.3 [8].
radio	The structure contains radio device (radio0, radio1, etc.) and antenna port for transmission of WSMs.
repeatRate	Repeat rate for messages as defined in 1609.3 [8] as number of messages per 5 sec intervals. Additionally, it can be set to 0 for transmitting a single message or “one shot” transmissions in PC5 instance.
payload	WSM message payload excluding message length field.

Specific details for each type definition are listed in the ASN.1 specification referenced in Appendix A.

8.1.1.4 StopWsmTx

This request is used to stop transmission of WSMs by the SUT. The WSM stream is identified by the RadioInterface and PSID.

```
StopWsmTx ::= SEQUENCE {
 psid         Psid Optional,
 radio         RadioInterface,
 ...
}
```

Table 5 StopWsmTx parameters

Parameters	**Explanation**
psid	Provider Service identifier as defined in 1609.3 [8].
radio	The structure contains radio device (radio0, radio1, etc) and antenna port for transmission of WSMs.

Specific details for each type definition are listed in the ASN.1 specification referenced in Appendix A.

8.1.1.5 StartWsmRx

This request is used to configure the SUT to receive messages and forward corresponding event indications to the TS. Information provided in this request can be used to invoke *WME-WSMService.request* and *WMEChannelService* from 1609.3[8].

```
StartWsmRx ::= SEQUENCE{

psid                       Psid OPTIONAL,
radio                      RadioInterface,
channelIdentifier          ChannelNumber80211 OPTIONAL,
timeSlot                   TimeSlot OPTIONAL,
eventHandling              EventHandling,
...,

destinationMACAddr         MACaddress OPTIONAL,

pduFilter                  OCTET STRING(SIZE(0..4)) OPTIONAL,

ssp                        SecurityPermission OPTIONAL

}
```

Table 8 StartWsmRx parameters

Parameters	Explanation
psid	Provider Service identifier as defined in 1609.3 [8].
radio	The structure contains radio device (radio0, radio1, etc) and antenna port for transmission of WSMs.
channelIdentifier	Channel number as defined in 1609.3 [8].
timeslot	Time slot or continuous channel usage as defined in 1609.3 [8].
eventHandling	Types of events which TS request to receive indications about. The types of events supported includes reception of a message, completion of message security verification, and etc.
DestinationMACAddr	L2 Destination address to be subscribed to or use wildcard for LTE V2X comms
pduFilter	Unique octets to be included at the beginning of the payload to be used for packet filtering
ssp	Secure Service Permission

Specific details for each type definition are listed in the ASN.1 specification referenced in Appendix A.

The SUT will send an *indication* message when it receives a WSM. Using *eventHandling* parameter, the TS can request to receive all WSMs or only those with matching PSID parameters. In the latter case, the PSID parameter is omitted.

The TS will expect to receive the *Indication* message within **50ms** after the corresponding WSM is received by the SUT.

8.1.1.6 StopWsmRx

This request is used to stop SUT reception of messages and generation of *indication* messages.

```
StopWsmRx ::= SEQUENCE{
   psid   Psid OPTIONAL,
   radio  RadioInterface,
   ...
}
```

Table 9 StopWsmRx parameters

Parameters	Explanation
psid	Provider Service Identifier as defined in 1609.3 [8].
radio	The structure contains radio device (radio0, radio1, etc.) and antenna port for receiving of WSMs.

If the preceding *StartWsmRx* omitted *psid* parameter, *psid* is omitted for the *StopWsmRx*.

Specific details for each type definition are listed in the ASN.1 specification referenced in the Appendix A.

8.1.1.7 StartWsaTxPerdiodic

This request is used to initiate transmission of WSA by the SUT. Information provided in this request can be used to invoke *WME-ProviderService.request* from 1609.3 [8]. WSAs will be sent as WSMs using the default PSID defined in 1609.3 [8].

```
StartWsaTxPerdiodic ::= SEQUENCE{
 radio                        RadioInterface,
 destinationMACAddr           MACaddress OPTIONAL,
 wsaChannelIdentifier         ChannelNumber80211 OPTIONAL,
 channelAccess                TimeSlot OPTIONAL,
 repeatRate                   RepeatRate OPTIONAL,
 ipService                    BOOLEAN,
 security
  SecurityContext (WITH COMPONENTS {
       contentType (mWSA)
    }),
 signatureLifetime            INTEGER(10...30000),
 infoElementIncluded          WaveElementsIncluded DEFAULT '000000000000000000000000'B,
 advertiserId                 AdvertiserIdentifier OPTIONAL,
 serviceInfos                 ServiceInfos OPTIONAL,
 channelinfos                 ChannelInfos OPTIONAL,
 wra                          RoutingAdvertisement OPTIONAL,
 dataRate                    DataRate80211 OPTIONAL,
userPriority                 UserPriority OPTIONAL,
transmitPowerLevel           TXpower80211 OPTIONAL,
 ...
}
```

Table 6 StartWsaTxPerdiodic parameters

Parameters	Explanation
radio	The structure contains radio device (radio0, radio1, etc) and antenna port for transmission of WSAs.
destinationMACAddr	Destination MAC address for the destination as defined in 1609.3 [8]. Default value set for broadcast transmissions.
wsaChannelIdentifier	Channel number to transmit WSAs as defined in 1609.3 [8].
channelAccess	Time slot or continuous channel usage as defined in 1609.3 [8].
repeatRate	Repeat rate for messages as defined in 1609.3 [8] as number of messages per 5 sec intervals. Additionally, it can be set to 0 for transmitting a single message.
ipService	Indicates if the WSA contains WRA for configuration of IP-based services
security	The structure security context including content type of payload (i.e., BSM, WSA, etc.) for selecting appropriate security profile; security type (i.e., unsecure, signed, etc.); optional reference to a certificate hashID.
signatureLifetime	Signature Lifetime as defined in 1609.3 [8].
infoElementsIncluded	A bit flag indicating which optional WAVE Info Elements included in the WSM-N-Header and into WSA message structure.
advertiserId	Advertiser Identifier as defined in 1609.3 [8].

serviceInfos	The structure containing sequence of service information elements as defined in 1609.3 [8].
channelinfos	The structure containing sequence of Channel Information elements as defined in 1609.3 [8].
wra	A structure containing WRA information. This field is required if ipService is set TRUE. Otherwise, it's omitted.
dataRate	Data Rate used for transmission of WSMs containing WSA. If omitted, use default value from the MIB
userPriority	User Priority used for transmission of WSMs containing WSA. If omitted, use default value from the MIB
transmitPowerLevel	Transmit Power setting used for transmission of WSMs containing WSA. If omitted, use default value from the MIB

Specific details for each type definition are listed in the ASN.1 specification referenced in Appendix A.

8.1.1.8 StopWsaTxPeriodic

This request is used to stop the current WSA transmissions by the SUT and delete associated provider services from the *ProviderServiceRequestTable*.

```
StopWsaTxPeriodic ::= SEQUENCE{
 radio         RadioInterface,
 ...
}
```

Table 7 StopWsaTxPeriodic parameters

Parameters	Explanation
radio	The structure contains radio device (radio0, radio1, etc.) and antenna port for transmission of WSAs.

Specific details for each type definition are listed in the ASN.1 specification referenced in the Appendix A.

8.1.1.9 AddWsaProviderService

This request is used to add a provider service and update WSA. The WSA must be started prior to this request using *StartWsaTxPerdiodic*.

```
AddWsaProviderService ::=SEQUENCE{
  radio         RadioInterface,
  serviceInfos  ServiceInfos,

  ...
}
```

Table 10 AddWsaProviderService

Parameters	Explanation
radio	The structure contains radio device (radio0, radio1, etc.) and antenna port for transmission of WSMs.
serviceInfos	The structure containing sequence of service information elements as defined in 1609.3 [8].

This request can add one or more service entries into an existing WSA. The new services must refer to already existing information in WSA such as Channel Info elements and WRA (if included).

Specific details for each type definition are listed in the ASN.1 specification referenced in Appendix A.

8.1.1.10 ChangeWsaProviderService

This request is used to change a provider service and updates WSA.

```
ChangeWsaProviderService ::= SEQUENCE{
     radio                 RadioInterface
     serviceInfos          ServiceInfos
     …
}
```

8.1.1.11 DelWsaProviderService

This request is used to remove a provider service and updates WSA. This request must only remove provider services previously added using *AddWsaProviderService*.

```
DelWsaProviderService ::=SEQUENCE{
 radio                           RadioInterface,
 serviceInfos                    ServiceInfos,
...
}
```

Table 11 DelWsaProviderService

Parameters	**Explanation**
radio	The structure contains radio device (radio0, radio1, etc) and antenna port for transmission of WSMs.
serviceInfos	The structure containing sequence of service information elements as defined in 1609.3 [8].

The *serviceInfo* structure must contain at least psid information for each service that will be removed.

Specific details for each type definition are listed in the ASN.1 specification referenced in Appendix A.

8.1.1.12 AddUserService

This request is used to add a user service to the SUT. Information provided in this request can be used to invoke *WME-UserService.request* and *WME-ChannelService* from 1609.3 [8].

```
AddUserService ::= SEQUENCE{ -- register user service via
 psid                            Psid,
 radio                           RadioInterface,
 userRequestType                 UserRequestType,
 wsaType                         WsaType,
 providerServiceContext          ProviderServiceContext OPTIONAL,
 channelIdentifier               ChannelNumber80211 OPTIONAL,
 sourceMACAddr                   MACaddress OPTIONAL,
 advertiserId                    AdvertiserIdentifier OPTIONAL,
 linkQuality                     INTEGER OPTIONAL,
 immediateAccess                 INTEGER(0..255) OPTIONAL,
 wsaChannelIdentifier            ChannelNumber80211 OPTIONAL,
 channelAccess                   TimeSlot OPTIONAL,
 evenHandling                    EventHandling,
...
reqPsidInSignCert                BOOLEAN OPTIONAL,

Ssp                              SecurityPermission OPTIONAL

}
```

Table 12 AddUserService parameters

Parameters	**Explanation**
psid	Provider Service identifier as defined in 1609.3 [8].

radio	The structure contains radio device (radio0, radio1, etc) and antenna port for transmission of WSMs.
userRequestType	User Request Type as defined in 1609.3 [8]. (options include autojoin on match, no service channel).
wsaType	WSA Type as defined in 1609.3 [8] (options include secure, unsecure).
providerServiceContext	Provider Service Context as defined in 1609.3 [8].
channelIdentifier	Channel number as defined in 1609.3 [8].
sourceMACAddr	Source MAC address as defined in 1609.3 [8].
advertiserId	Advertiser ID as defined in 1609.3 [8].
linkQuality	Link Quality as defined in 1609.3 [8].
immediateAccess	Channel Load as defined in 1609.3 [8].
wsaChannelIdentifier	Channel number to transmit WSAs as defined in 1609.3 [8].
channelAccess	Channel to listen for WSA
eventHandling	Event Handling when service is joined
reqPsidInSignCert	Support for Require PSID in Signing Certificate
Ssp	Security Specific Permissions to be checked when messages /service is received

Specific details for each type definition are listed in the ASN.1 specification referenced in Appendix A.

8.1.1.13 DelUserService

This request is used to delete a user service on the SUT previously requested by the *AddUserService* request.

```
DelUserService ::= SEQUENCE{
 psid                        Psid,
 radio                RadioInterface,  ...
}
```

Table 13 DelUserRequestService parameters

Parameters	Explanation
psid	Provider Service identifier as defined in 1609.3 [8].
radio	The structure contains radio device (radio0, radio1, etc) and antenna port for transmission of WSMs.

Specific details for each type definition are listed in the ASN.1 specification referenced in Appendix A.

8.1.2 Content Type

The *TCI-wsm* module specifies content types. There are 7 content types in the TCI-wsm

```
ContentType ::= ENUMERATED {
mOther         (0),
mIeee16092Data (1),
mWSA           (2),
mBSM           (3),
mMAP           (4),
mSPAT          (5),
mTIM           (6)
}
```

8.1.3 Signer Identifier Type

The TCI-wsm module specifies signer identifier types. There are 4 signer identifier types in the TCI-wsm.

```
SignerIdentifierType ::= ENUMERATED {
unsecure                     (0),
useSecProfilePerContentType  (1),
signIncludeCertificate       (2),
signInlcudeDigest            (3),
}
```

8.2 TCI-ip module

The *TCI-ip* module defines request messages from the TS to the SUT to trigger transmission and/or reception of messages using IPv6-based protocols. It also includes messages for retrieving IPv6 address information from the SUT.

8.2.1 Base Classes

8.2.1.1 IPv6TxRecord

This base class specifies ……

```
Ipv6TxRecord ::= SEQUENCE{
   Radio              radioInterface
   interfaceName      UTF8STRING(SIZE(1..255)),
   destIpAddress      IPv6Address,
   destPort           ServicePort OPTIONAL
   protocol           ENUMERATED {tcp, udp, icmp},
   repeatRate         RepeatRate OPTIONAL,
   eventHandling      EventHandling (WITH COMPONENTS {…, eventFlag}) OPTIONAL
   payload            Opaque(SIZE(0..ipMtu)) OPTIONAL,
   …
}
```

8.2.1.2 IPv6RxRecord

This base class specifies ……

```
IPv6RxRecord ::= SEQUENCE{
   radio              RadioInterface,
   interfaceName      UTF8String(SIZE(1...255))
   listenPort         ServicePort
   Protocol           ENUMERATED {tcp (0), udp (1)},
   eventHandling      EventHandling (WITH COMPONENTS {…, eventFlag({eIpv6PktRx})}) OPTIONAL,
   …
}
```

8.2.2 Request messages

8.2.2.1 GetIpv6InterfaceInfo

This request is used to retrieve IPv6 configuration from the SUT. This message uses a service provided by the IP domain.

```
GetIPv6InterfaceInfo ::= SEQUENCE{
       radio          RadioInterface (WITH COMPONENTS {..., antenna ABSENT }),
       ...
}
```

Table 14 getIPv6InterfaceInfo Message Parameters

Parameters	Explanation
radio	The structure contains radio device (radio0, radio1, etc) and antenna port for transmission of IPv6 packets.

The requested IPv6 configuration is returned in the *ResponseInfo* message found in the *TCI-responseInfo* module and contains:

```
Ipv6InterfaceInfo ::= SEQUENCE OF SEQUENCE {
       interfaceName  UTF8String(SIZE(1..255)), ,
        ipAddress     SEQUENCE OF IPv6Address,
        macAddress    MACaddress,
        defaultGateway       IPv6Address OPTIONAL,
        primaryDns           IPv6Address OPTIONAL,
        gatewayMacAddress    MACaddress OPTIONAL,
 ...
}
```

8.2.2.2 SetIpv6Address

This request is used to change SUT IPv6 configuration.

```
SetIPv6Address ::= SEQUENCE{
        radio         RadioInterface (WITH COMPONENTS {..., antenna ABSENT}),
       interfaceName  UTF8String(SIZE(1...255)),
       ipAddress      IpAddress OPTIONAL,

         ...
}
```

Table 15 setIPv6Address Message Parameters

Parameters	Explanation
radio	The structure contains radio device (radio0, radio1, etc) and antenna port for transmission of IPv6 packets.
interfaceName	Interface Name is an identifier of the interface provided by the SUT in response to the *GetIpv6InterfaceInfo.*
ipAddress	IPv6 address specified in canonical format (e.g. 2001:ff::1) to be assigned to the interface. If omitted, the SUT must assign a randomly chosen IPv6 address.

8.2.2.3 StartIPv6Tx

This request is used to initiate transmission of IPv6 packets by the SUT.

```
StartIPv6Tx ::= IPv6TxRecord(WITH COMPONENTS{
 Radio ( WITH COMPONENTS {…, antenna ABSENT})
 interfaceName
 destIpAddress
 destPort
 protocol
 repeatRate OPTIONAL
 eventHandling OPTIONAL
 payload PRESENT
})
```

Table 16 startIPv6Tx Message Parameters

Parameters	Explanation

radio	The structure contains radio device (radio0, radio1, etc) and antenna port for transmission of IPv6 packets.
interfaceName	Interface Name is an identifier of the interface provided by the SUT in response to the *GetIpv6InterfaceInfo.*
destipAddresses	Destination host IPv6 address specified in canonical format (e.g. 2001:ff::1).
destPort	Destination host port used for the reception of IPv6 packets.
Protocol	IP protocol : tcp, udp or icmp.
repeatRate	Repeat rate for messages as defined in 1609.3. Additionally, can be set to 0 for transmitting a single message.
eventHandling	This parameter is omitted any protocol except icmp – see SendIpv6Ping.
payload	The contents of the message.

8.2.2.4 StopIPv6Tx

This request is used to cease transmission of IPv6 packets by the SUT.

```
StopIPv6Tx ::= StartIPv6Tx (WITH COMPONENTS {
   radio (WITH COMPONENTS {…, antenna ABSENT}),
   interfaceName
   destIpAddress,
   destPort
   protocol,
   repeatRate ABSENT,
   eventHandling ABSENT
   Payload ABSENT
})
```

See Table 16 for an explanation.

8.2.2.5 StartIpv6Ping

This request is used to transmit a single ping message, or a multiple ping messages from the SUT over IPv6 and receive ping echo from the remote host.

```
StartIPv6Ping ::= IPv6TxRecord (WITH COMPONENTS{
   radio,
   interfaceName,
   destIpAddress,
   destPort ABSENT
   protocol (icmp),
   repeatRate OPTIONAL,
   eventHandling   (WITH COMPONENTS {…, eventFlag ({eIcmp6PktRx})}),
   Payload ABSENT
})
```

Table 17 sendIPv6Ping Message Parameters

Parameters	**Explanation**
radio	The structure contains radio device (radio0, radio1, etc) and antenna port for transmission of ping v6 messages.
interfaceName	Interface Name is an identifier of the interface provided by the SUT in response to the *GetIpv6InterfaceInfo.*
destipAddresses	Destination host IPv6 address specified in canonical format (e.g. 2001:ff::1).
destPort	Omitted
protocol (icmp),	The protocol used for the ping (ICMP in this case).

repeatRate	Repeat rate for messages as defined in 1609.3 as number of messages per 5 sec intervals. Additionally, it can be set to 0 for transmitting a single message.
eventHandling	A parameter is used to request SUT to send an *indication* to the TS when ping echo is received.
payload	No payload is required for this message.

8.2.2.6 StopIPv6Ping

```
StopIPv6Ping ::= IPv6TxRecord (WITH COMPONENTS{
   Radio (WITH COMPONENTS {…, antenna ABSENT}),
   interfaceName,
   destIpAddress,
   destPort ABSENT
   protocol (icmp),
   repeatRate ABSENT,
   eventHandling ABSENT
   Payload ABSENT
})
```

See Table 17 for an explanation.

8.2.2.7 StartIPv6Rx

This request is used to initiate reception of IPv6 packets by the SUT.

```
StartIPv6Rx ::= IPv6RxRecord(WITH COMPONENTS{
 Radio
 interfaceName
 listenPort
 protocol
 eventHandling ABSENT
})
```

Table 18 startIPv6Rx Message Parameters

Parameters	**Explanation**
radio	The structure contains radio device (radio0, radio1, etc) and antenna port for reception of IPv6 packets.
interfaceName	Interface Name is an identifier of the interface provided by the SUT in response to the *GetIpv6InterfaceInfo.*
listenPort	The port number the SUT should use to listen to IPv6 packets.
protocol	The protocol used for the reception (TCP or UDP).
eventHandling	A parameter is used to request SUT to send an *indication* to the TS when an IPv6 packet is received.

8.2.2.8 StopIPv6Rx

This request is used to cease reception of IPv6 packets by the SUT.

```
StopIPv6Rx ::= IPv6RxRecord(WITH COMPONENTS{
 Radio ( WITH COMPONENTS {…, antenna ABSENT})
 interfaceName
 listenPort
 protocol
 eventHandling ABSENT
})
```

Table 19 stopIPv6Rx Message Parameters

Parameters	**Explanation**

radio	The structure contains radio device (radio0, radio1, etc.) and antenna port for reception of IPv6 packets.
interfaceName	Interface Name is an identifier of the interface provided by the SUT in response to the *GetIpv6InterfaceInfo.*
listenPort	The port number the SUT should use to listen to IPv6 packets.
protocol	The protocol used for the reception (TCP or UDP).
eventHandling	Not required.

8.3 Response, ResponseInfo, Indication and Exception messages

8.3.1 Response messages

The *Response* message is sent in response to the *Request*. It is defined in the *TCI-CommonType* module. A *Response* message must be triggered within **50ms** after an SUT received a *Request* message. If no *Response* is received, the TS will attempt to re-initialize the SUT or may request user assistance.

```
Response ::= SEQUENCE {
 msgID        MsgID,
 resultCode  ResultCode,
 exception   Exception OPTIONAL,
 ...
}
```

Table 20 Response message

Parameters	**Explanation**
msgID	Use the same MsgID from the corresponding *Request* message. msgIDs are listed in the Table 31.
resultCode	Success or Failure enumerated as 0 or 1 respectively.
exception	This parameter contains additional information if exception must be reported to the TS (i.e. failure, warning, etc.). See details in 7.3.4.

Specific details for each type definition are listed in the ASN.1 specification referenced in Appendix A.

8.3.2 Indication messages

The *Indication* message is sent from the SUT to TS. It is defined in the *TCI-indication* module.

```
Indication ::= SEQUENCE{
 radio             RadioInterface (WITH COMPONENTS {…, antenna ABSENT}),
 event             Event,
 eventParams       EventParams OPTIONAL,
 pdu               Pdu OPTIONAL,
 exception         Exception OPTIONAL,
 ...
}
```

Table 21 Indication message

Parameters	**Explanation**
radio	The structure contains radio device (radio0, radio1, etc.) and antenna port for transmission of WSAs.
event	Enumerated list of events that when occur, will generate an Indication messages.

eventParams	Event parameters contain some data related to message reception but not included in the message payload (e.g. message RCPI).
pdu	Optional element containing payload of the message identified by the event.
exception	Optional element which is used to report exception. It is omitted if no exception is reported.

Specific details for each type definition are listed in the ASN.1 specification referenced in Appendix A.

Table 22 lists event types that may trigger transmission of an *Indication* message. Those event types are defined in the *TCI-Indication* module.

Table 22 Events which can trigger Indication messages

Parameters	Enumerated	Explanation
e80211PktRx	1	SUT received an inbound 802.11 frame
e16093PktRx	2	SUT received an inbound 1609.3 packet
eWsmPktRx	3	SUT received an inbound WSM (with matching PSID)
eIpv6PktRx	4	SUT received an inbound IPv6 frame over DSRC
eIcmp6PktRx	5	SUT received an inbound ping (ICMP) IPv6 echo message
eIpv6ConfigChanged	6	SUT IPv6 address change on one of the DSRC radio interfaces
eDot3ChannelAssigned	7	SUT assigned a channel as per WME-Notification.indication
eDot3RequestMatchedAvailAppService	8	request matched with available application-service as per WMENotification.indication
eDot2VerificationCompleteWithResult	9	Inbound WSM or WSA message signature verification is complete
exception	15	SUT generated an exception.

8.3.3 ResponseInfo messages

This message is used to retrieve configuration information from the SUT. It is defined in the *TCI-responseInfo* module. A *ResponseInfo* message must be triggered within **50ms** after an SUT received a *Request* message. If no *ResponseInfo* is received, the TS will attempt to re-initialize the SUT or may request user assistance.

```
ResponseInfo ::= SEQUENCE {
    msgID              MsgID,
    resultCode         ResultCode,
    info               InfoContent OPTIONAL,
    exception          Exception OPTIONAL,
    ...
}
```

Table 23 ResponseInfo message

Parameters	Explanation
MsgID	Use the same MsgID from the corresponding *Request* message.
resultCode	Success or Failure enumerated as 0 or 1 respectively.
info	This parameter contains information requested from the SUT. If SUT detects an error which prevents it to report the requested information, then info parameter is omitted and instead exception parameter is included.
exception	This optional parameter is included SUT must report exception explaining the possible details of the failure result code. See details in 7.3.4.

Specific details for each type definition are listed in the ASN.1 specification referenced in Appendix A.

8.3.4 Exception messages

Exception is a message sent from the SUT to the TS. It is used to report certain conditions to the TS. There are no exception messages from the TS to the SUT. Upon reception of an Exception message, the TS does not need to send a response back to the SUT.

The SUT sends each exception only once and does not need to repeat it. The SUT does not send an exception cancellation if the condition causing exception stops. If repeated exceptions occur due to repeatable events, e.g. reception of invalid message from the TS, then one Exception message is sent for every event which generates an exception.

An Exception message must be triggered within **50ms** after the corresponding event occurred on the SUT.

Exception information can also be reported in the *Response*, *Indication* and *ResponseInfo*. Then, the TS does not need to send a standalone exception message. It is defined in the *TCI-CommonTypes* module.

```
Exception ::= SEQUENCE{
   type         ExceptionType,
   id           ExceptionId OPTIONAL,
   module       Module OPTIONAL,
   description  Exception OPTIONAL
...
}
```

Table 24 Exception message

Parameters	Explanation
type	Can be info, warning or error.
id	Integer identifier assigned for the exception.
module	A text string providing the name of a module where exception is detected.
description	This parameter contains a text string describing the exception.

Specific details for each type definition are listed in the ASN.1 specification referenced in Appendix A.

Table 25 Defined exceptions

id	Type	Description
1	error	Critical error
2	error	Incorrect parameter value
3	error	Missing parameter
4	error	Radio interface is unavailable

9 TCI frames

9.1 TCI80211 frame

9.1.1 Supported use cases

Use cases (UC) supported by **TCI80211** are listed in the Table 26.

Note, in the Message Sequence column, the common prefix *TCIMsg.frame* is omitted. For example, the full name for *request.SetInitialState* is *TCIMsg.frame.request.SetInitialState*.

Table 26 Use cases supported by TCI802.11

UC #	Use case objective	Flow Direction	Message Sequence
1	Reset the SUT to the Initial state	TS → SUT SUT → TS	request.SetInitialState response
2	Set the WSM Transmission Info	TS → SUT SUT → TS	request.setWsmTxInfo
3	The SUT transmits a single or periodic WSMs	TS → SUT SUT → TS	request. StartWsmTx response
4	The SUT stops transmitting periodic WSMs	TS → SUT SUT → TS	request. StopWsmTx response
5	The SUT receives WSMs and sends event indications to the TS	TS → SUT SUT → TS	request. StartWsmRx response
6	The SUT stops receiving WSMs	TS → SUT SUT → TS	request. StopWsmRx response

9.1.2 *Request* Messages

Table 27 lists all supported *Request* messages supported in the ***TCI16093*** frame. When SUT sends a *Response* message, it must include the *MsgID* and Type corresponding to the *Request* message.

Most of these messages are imported from the common *TCI-wsm* module.

Table 27 Listing of *Request* messages

Request Messages Type	MsgID	Explanation
SetInitialState	1	Request to configure SUT to the Initial state
Dot11SetWsmTxInfo	2	Request to configure the WSM transmission information
Dot11StartWsmTx	3	Request to start transmission of WSMs
StopWsmTx	4	Request to stop transmission of WSMs
StartWsmRx	5	Request to start reception of WSMs
StopWsmRx	6	Request to stop reception of WSMs

9.1.2.1 SetInitialState

This request is used to set the SUT in initial condition. This request is defined in the *TCI-wsm* module.

9.1.2.2 Dot11SetWsmTxInfo

This request is used to configure device parameters before transmitting WSMs per 80211. This request correlates to the pre-defined *setWsmTxInfo* request in the *TCI-wsm* module.

```
Dot11SetWsmTxInfo ::= SetWsmTxInfo (WITH COMPONENTS {
     psid,
     radio,
     security,
     transmitPowerLevel,
     infoElementsIncluded,
     userPriority,
     channelIdentifier,
     dataRate,
     timeslot,
     repeatRate,
     destinationMACAddr (‘FFFFFFFFFFFF’H),
     channelLoad           ABSENT,
     expiryTime            ABSENT,
     payload               ABSENT
})
```

9.1.2.3 *Dot11StartWsmTx*

This request is used to initiate transmission of WSMs by the SUT. This request correlates to the pre-defined *StartWsmTx* request in the *TCI-wsm module*.

```
Dot11StartWsmTx ::= StartWsmTx (WITH COMPONENTS {
                 psid,
                 radio,
                 repeatRate,
                 Payload(SIZE(0..dsrcMtu)) PRESENT
})
```

9.1.2.4 *StopWsmTx*

This request is used to cease transmission of WSMs by the SUT. This request is defined in the *TCI-wsm* module.

9.1.2.5 *StartWsmRx*

This request is used to configure the SUT to receive messages and forward corresponding event indications to the TS. This request is defined in the *TCI-wsm* module.

9.1.2.6 *StopWsmRx*

This request is used to stop the SUT reception of messages and generation of *Indication* messages. This request is defined in the *TCI-wsm* module.

9.1.3 *Response* messages

The *Response* message is sent in response to the *Request*. *Response* is defined in the *TCI-CommonTypes*

module.

9.1.4 *Indication* messages

The *Indication* message is sent from the SUT to the TS indicating an occurrence of a predefined event. *TCI-80211* defines *Dot11Indication* as follows:

```
Dot11Indication ::= Indication (WITH COMPONENTS {
radio,
event (e80211PktRx),
eventParams (WITH COMPONENTS {d80211frame} ) OPTIONAL,
pdu OPTIONAL,
exception OPTIONAL
})
```

where *Indication* is defined in the *TCI-indication* module.

9.1.5 *Exception* messages

Exception is a message sent from the SUT to the TS. It is used to report exception conditions to the TS. *Exception* is defined in the *TCI-CommonTypes* module.

9.2 TCI16094 frame

9.2.1 Supported use cases

Use cases supported by **TCI16094** are listed in Table 28.

Table 28 Use cases supported by TCI16094

UC #	Use case objective	Flow Direction	Message Sequence
1	Reset the SUT to the Initial state	TS → SUT SUT → TS	request.SetInitialState response
2	To configure the SUT WSM transmit parameters such as psid, radio, channel, timeslot, data rate … etc.	TS → SUT SUT → TS	request. SetWsmTxInfo response
3	The SUT transmits a single or periodic WSMs	TS → SUT SUT → TS	request. StartWsmTx response
4	The SUT stops transmitting periodic WSMs	TS → SUT SUT → TS	request. StopWsmTx response
5	The SUT receives WSMs and sends event indications to the TS	TS → SUT SUT → TS	request. StartWsmRx response
6	The SUT stops receiving WSMs	TS → SUT SUT → TS	request. StopWsmRx response
7	Add Transmit Profile of SUT	TS → SUT SUT → TS	request.AddTxProfile response
8	Delete Transmit Profile of SUT	TS → SUT SUT → TS	request.DelTxProfile response
9	The TS requests information from the SUT about the radio (0..3) used for IPv6 Communication	TS → SUT SUT → TS	request.GetIpv6InterfaceInfo response
10	The SUT transmits single or periodic IPv6 packets	TS → SUT SUT → TS	request. StartIPv6Tx response
11	The SUT stops transmitting periodic IPv6 packets	TS → SUT SUT → TS	request. StopIPv6Tx response
12	The SUT receives IPv6 packets and sends event indications to the TS	TS → SUT SUT → TS	request. StartIPv6Rx response
13	The SUT stops receiving IPv6 packets	TS → SUT SUT → TS	request. StopIPv6Rx response
14	The SUT to configure its radio, interface name and IPv6 address used to transmit and receive IPv6 packets	TS → SUT SUT → TS	request.SetIpv6 Ipv6Address response

15	The SUT to ping another IPv6 device specifying the radio, the interface, destination IPv6 address and port to use for the transmission and reception. Received ping echo is forwarded to the TS	TS → SUT SUT → TS	request. StartIpv6Pingresponse
16	The SUT stops transmission of ping to another IPv6 device	TS → SUT SUT → TS	Request StopIPv6Ping response

9.2.2 *Request* Messages

Table 29 lists all supported R*equest* messages supported in the *TCI-16094* frame. When the SUT sends a *Response* message, it must include the *MsgID* corresponding to the *Request* message.

Table 29 Listing of *Request* messages

Request Messages	MsgID	Explanation
SetInitialState	1	Request to configure SUT to the Initial state
SetWsmTxInfo	2	Request to configure WSM transmit parameters
StartWsmTx	3	Request to start transmission of WSMs
StopWsmTx	4	Request to stop transmission of WSMs
StartWsmRx	5	Request to start reception of WSMs
StopWsmTx	6	Request to stop reception of WSMs
GetIpv6InterfaceInfo	7	The TS requests IPv6 configuration from the SUT
SetIpv6Address	8	The TS requests the SUT to change its IPv6 configuration
StartIPv6Tx	9	Request to start transmission of IPv6 packets
StopIPv6Tx	10	Request to stop transmission of IPv6 packets
StartIPv6Rx	11	Request to start reception of IPv6 packets
StopIPv6Rx	12	Request to stop reception of IPv6 packets
StartIpv6Ping	13	Transmit ping messages over IPv6 and receive ping echo from the remote host
StopIpv6Ping	14	Stop transmission of ping messages over IPv6
AddTxProfile	15	Add transmission profile for IPv6 testing without WSA
DelTxProfile	16	Delete transmission profile for Ipv6 testing without WSA

9.2.2.1 SetInitialState

This request is used to set the SUT in initial condition. This request is defined in the *TCI-wsm* module.

9.2.2.2 Dot4SetWsmTxInfo

This request is used to configure the SUT's WSM transmission parameters. This request correlates to the pre-defined *setWsmTxInfo* request in the *TCI-wsm* module.

9.2.2.3 Dot4StartWsmTX
This request is used to initiate transmission of WSMs by the SUT. This request correlates to the pre-defined *startWsmTx* request in the *TCI-wsm* module.

9.2.2.4 StopWsmTx
This request is used to cease transmission of WSMs by the SUT. This request is defined in the *TCI-wsm* module.

9.2.2.5 StartWsmRX
This request is used to configure the SUT to receive messages and forward corresponding event indications to the TS. This request is defined in the *TCI-wsm* module.

9.2.2.6 StopWsmRX
This request is used to stop SUT reception of messages and generation of *indication* messages. This request is defined in the *TCI-wsm* module.

9.2.2.7 GetIpv6InterfaceInfo
This request is used to requests IPv6 configuration from the SUT. This request is defined in the *TCI-ip* module.

9.2.2.8 SetIpv6Address
This request is used to change SUT IPv6 configuration. This request is defined in the *TCI-ip* module.

9.2.2.9 StartIPv6Tx
This request is used to initiate transmission of IPv6 packets by the SUT. This message uses a service provided by the IP domain. Please refer to section 7.2.1.3 for additional information.

9.2.2.10 StopIPv6Tx
This request is used to cease transmission of IPv6 packets by the SUT. This request is defined in the *TCI-ip* module.

9.2.2.11 StartIPv6Rx
This request is used to initiate reception of IPv6 packets by the SUT. This request is defined in the *TCI-ip* module.

9.2.2.12 StopIPv6Rx
This request is used to cease reception of IPv6 packets by the SUT. This request is defined in the *TCI-ip* module.

9.2.2.13 StartIpv6Ping
This request is used to transmit a single ping message from the SUT over IPv6 and receive a ping echo from the remote host. This request is defined in the *TCI-ip* module.

9.2.2.14 StopIpv6Ping
This request stops transmission of ping messages from the SUT over IPv6. This request is defined in the *TCI-ip* module.

9.2.2.15 AddTxProfile
This request adds transmissions profile of the SUT for IPV6 Testing without WSAs. This request is defined in the *TCI-ip* module.

9.2.2.16 DelTxProfile
This request deletes the transmission profile of the SUT for IPV6 Testing without WSAs. This request is defined in the *TCI-ip* module.

9.2.3 *Response* messages

The *Response* message is sent in response to the *Request*. *Response* is defined in the *TCI-CommonTypes* module.

9.2.4 *Indication* messages

The *Indication* message is sent from the SUT to the TS indicating an occurrence of a predefined event. *TCI-16094* defines *Dot4Indication* as follows:

```
Dot4Indication ::= Indication (WITH COMPONENTS{
 radio,
 event (e16093PktRx|
        eWsmPktRx |
        eIpv6PktRx |
        eIcmp6PktRx |
        eIpv6ConfigChanged |
        eDot3ChannelAssigned |
        eDot3RequestMatchedAvailAppService |
        exception),
 eventParams (WITH COMPONENTS {service} |
              WITH COMPONENTS {wsm} |
              WITH COMPONENTS {ip}
              ) OPTIONAL
 pdu OPTIONAL
 exception OPTIONAL
})
```

where *Indication* is defined in *TCI-indication* module.

9.2.5 *ResponseInfo* messages

This message is used to retrieve configuration information from SUT.

```
Dot4ResponseInfo ::= ResponseInfo (WITH COMPONENTS{
msgId,
resultCode
info (WITH COMPONENTS {ipv6InterfaceInfo} ) OPTIONAL,
exception OPTIONAL
})
```

9.2.6 *Exception* messages

Exception is a message sent from the SUT to the TS. It is used to report exception conditions to the TS. *Exception* is defined in the *TCI-CommonTypes* module.

9.3 TCI16093DSRC frame

9.3.1 Supported use cases

Use cases (UC) supported by TCI16093DSRC are listed in Table 30.

Note, in the Message Sequence column, the common prefix *TCIMsg.frame* is omitted. For example, the full name for *request.SetInitialState* is *TCIMsg.frame.request.SetInitialState*.

Table 30 Use cases supported by TCI16093

UC #	Use case objective	Flow Direction	Message Sequence
1	Reset the SUT to the Initial state	TS → SUT SUT → TS	request.SetInitialState response
2	The SUT transmits a single or periodic WSMs	TS → SUT SUT → TS TS → SUT SUT → TS	request.SetWsmTxInfo response request.StartWsmTx response
3	The SUT stops transmitting periodic WSMs	TS → SUT SUT → TS	request.StopWsmTx response
4	The SUT receives WSMs and sends event indications to the TS	TS → SUT SUT → TS SUT → TS	request.StartRx response indication
5	The SUT stops receiving WSMs	TS → SUT SUT → TS	request.StopRx response
6	The SUT starts transmitting WSAs	TS → SUT SUT → TS	request.StartWsaTxPeriodic response
7	The SUT stops transmitting WSAs	TS → SUT SUT → TS	request.StopWsaTxPeriodic response
8	The SUT adds a provider service to WSA	TS → SUT SUT → TS	request.AddWsaProviderService response
9	The SUT deletes a provider service from WSA	TS → SUT SUT → TS	request.DelWsaProviderService response
10	The SUT registers a user service and notifies the TS when it is activated	TS → SUT SUT → TS SUT → TS	request.AddUserService response indication
11	The SUT removes a registered user service	TS → SUT SUT → TS	request.DelUserService response
12	The TS requests IPv6 configuration from the SUT	TS → SUT SUT → TS	request. GetIpv6InterfaceInfo responseInfo
13	The TS requests the SUT to change its IPv6 configuration	TS → SUT SUT → TS	request.SetIpv6Address response
14	Transmit a single ping message over IPv6 and receive ping echo from the remote host	… TS → SUT SUT → TS SUT → TS	Start with Use Case 10, then request.SendIpv6Ping response indication
15	An exception occurred on SUT and reported to the TS	SUT → TS	exception
16	SUT joins a WSA and transmits WSMs on a Service channel	…	Run Use Case 10 Wait for the indication message and do Use Case 2

17	SUT joins a WSA and receives WSMs on a Service channel		Run Use Case 10 Wait for the indication message and do Use Case 4

The following dependencies are established among use cases:

- UC1 must precede UC 2, UC4, UC6, UC10, UC12, UC13, UC14
- UC3 must follow UC2 UC5 must follow UC4 UC7 must follow UC6 UC8 must follow UC6
- UC9 must follow UC8
- UC11 must follow UC10
- UC12, UC13, UC14 may follow in any order
- UC15 may occur at any time, including during execution of any other UC.

9.3.2 *Request* messages

Table 31 lists all supported R*equest* messages supported in the *TCI16093* frame. When the SUT sends a *Response* message, it must include the MsgID corresponding to the *Request* message.

Table 31 Listing of *Request* messages

Request Messages	MsgID	Explanation
SetInitialState	1	Request to configure SUT to the Initial state
SetWsmTxInfo	2	Request to set parameters used for transmissions of WSMs
StartWsmTx	3	Request to start transmission of WSMs
StopWsmTx	4	Request to stop transmission of WSMs
StartWsaTxPerdiodic	5	Request to start transmission of WSAs
StopWsaTxPeriodic	6	Request to stop transmission of WSAs
StartWsmRx	7	Request to start receiving WSMs
StopWsmRx	8	Request to stop receiving WSMs
AddWsaProviderService	9	Request to add a service provider to an existing WSA broadcast
ChangeWsaProviderService	10	Request to change a service provider to an existing WSA broadcast
DelWsaProviderService	11	Request to delete a service provider from an existing WSA broadcast
AddUserService	12	Request to add a user service
DelUserService	13	Request to delete a user service
GetIpv6InterfaceInfo	14	Request to SUT to report its IPv6 configuration
SetIpv6Address	15	Request to SUT to set its IPv6 address
StartIpv6Ping	16	Request to SUT to send a ping (ICMP over IPv6)
StopIpv6Ping	17	Request to SUT to stop sending ping (ICMP over IPv6)

9.3.2.1 SetInitialState

This request is used to set the SUT in initial condition. This request is defined in the *TCI-wsm* module.

9.3.2.2 Dot3SetWsmTxInfo
This request is used to configure the SUT's WSM transmission parameters. This request correlates to the pre-defined *SetWsmTxInfo* request in the *TCI-wsm* module.

9.3.2.3 Dot3StartWsmTx
This request is used to initiate transmission of WSMs by the SUT. Information about the expected content of the WSM needs to be set via the 8.3.2.2 Dot3SetWsmTxInfo request before. This request correlates to the pre-defined *StartWsmTx* request in the *TCI-wsm* module.

9.3.2.4 StopWsmTx
This request is used to cease transmission of WSMs by the SUT. This request is defined in the *TCI-wsm* module.

9.3.2.5 StartWsaTxPerdiodic
This request is used to initiate transmission of WSA by the SUT. This request is defined in the *TCI-wsm* module.

9.3.2.6 StopWsaTxPeriodic
This request is used to stop the current WSA transmissions by the SUT and delete associated provider services from the *ProviderServiceRequestTable*. This request is defined in the *TCI-wsm* module.

9.3.2.7 StartWsmRx
This request is used to configure the SUT to receive messages and forward corresponding event indications to the TS. This request is defined in the *TCI-wsm* module.

9.3.2.8 StopWsmRx
This request is used to stop the SUT's reception of messages and generation of *indication* messages. This request is defined in the *TCI-wsm* module.

9.3.2.9 AddWsaProviderService
This request is used to add a provider service and update WSA. This request is defined in the *TCI-wsm* module.

9.3.2.10 changeWsaProviderService
This request is used to change a provider service and update WSA. This request is defined in the TCI-wsm module.

9.3.2.11 DelWsaProviderService
This request is used to removes a provider service and updates WSA. This request is defined in the *TCI-wsm* module.

9.3.2.12 AddUserService
This request is used to add a user service to the SUT. This request is defined in the *TCI-wsm* module.

9.3.2.13 DelUserService
This request is used to delete a user service on the SUT previously requested by the *AddUserService* request. This request is defined in the *TCI-wsm* module.

9.3.2.14 GetIpv6InterfaceInfo
This request is used to retrieve IPv6 configuration from the SUT. This request is defined in the *TCI-ip* module.

9.3.2.15 SetIpv6Address
This request is used to set IPv6 address on the SUT. This request is defined in the *TCI-ip* module.

9.3.2.16 StartIpv6Ping

This request is used to request the SUT to transmit a single ping message over IPv6 and receive a ping echo from the remote host. This request is defined in the *TCI-ip* module.

9.3.2.17 StopIpv6Ping

This request is used to stop requesting the SUT to transmit ping messages. This request is defined in the TCI-ip module.

9.3.3 *Response* messages

The *Response* message is sent in response to the *Request*. *Response* is defined in the *TCI-CommonTypes*

module.

9.3.4 *Indication* messages

The *Indication* message is sent from the SUT to the TS indicating an occurrence of a predefined event. *TCI-16093* defines *Dot3Indication* as follows:

```
Dot3Indication ::= Indication (WITH COMPONENTS{
 radio,
 Event ( e16093PktRx |
        eWsmPktRx |
                 eIpv6PktRx |
                 eIcmp6PktRx |
                 eIpv6ConfigChanged |
                 eDot3ChannelAssigned |
                 eDot3RequestMatchedAvailAppService |
                 eDot2VerificationCompleteWithResult |
                 exception),
 eventParams (WITH COMPONENTS {service} |
             WITH COMPONENTS {wsm} |
              WITH COMPONENTS {ip}
             WITH COMPONENTS {security} |
             )OPTIONAL,
 pdu OPTIONAL,
 exception OPTIONAL
})
```

where *Indication* is defined in the *TCI-indication* module.

Table 32 Indication message

Parameters	Explanation
radio	The structure contains radio device (radio0, radio1, etc) and antenna port for transmission of WSAs.
event	Enumerated list of events that when occur, will generate an Indication messages. See 7.3.2 for the list of pre-defined events.
eventParams	Event parameters contain some data related to message reception but not included in the message payload.
pdu	Optional element containing payload of the message identified by the event.
exception	Optional element which is used to report exception. It is included if an exception is reported.

The SUT does not need to send both an *Indication* message with an *exception* parameter and a separate *Exception* message. If the SUT detects an exception, which doesn't not prevent it to receive and process subsequent messages, the SUT must report the exception in the *Indication* message. The SUT must use the *Exception* message if the exception condition causes the SUT to abort generation of *Indication* messages.

Specific details for each type definition are listed in the ASN.1 specification referenced in Appendix A.

9.3.5 *ResponseInfo* messages

This message is used to retrieve configuration information from the SUT.
TCI-16093 defines *Dot3ResponseInfo* as follows:

```
Dot3ResponseInfo ::= ResponseInfo (WITH COMPONENTS{
 msgID,
 resultCode,
 info (WITH COMPONENTS {ipv6InterfaceInfo} |
      WITH COMPONENTS {sutInfo} |
      WITH COMPONENTS {pktCount}) OPTIONAL,
 exception OPTIONAL
})
```

Table 33 ResponseInfo message

Parameters	Explanation
msgID	Use the same MsgID from the corresponding *Request* message. MsgIDs are listed in the Table 31.
resultCode	Success or Failure enumerated as 0 or 1 respectively.
info	This parameter contains information requested from the SUT. If SUT detects an error which prevents it to report the requested information, then info parameter is omitted and instead exception parameter is included.
exception	This optional parameter is included if SUT must report exception explaining the possible details of the Failure result code. See details in 8.3.6..

Specific details for each type definition are listed in the ASN.1 specification referenced in Appendix A.

9.3.6 *Exception* messages

Exception is a message sent from the SUT to the TS. It is used to report exception conditions to the TS. *Exception* and defined in the *TCI-CommonTypes* module.

9.4 TCI16093PC5 frame

9.4.1 Supported use cases

Use cases (UC) supported by TCI16093PC5 are listed in Table 40.

Note, in the Message Sequence column, the common prefix *TCIMsg.frame* is omitted. For example, the full name for *request.SetInitialState* is *TCIMsg.frame.request.SetInitialState*.

Table 40 Use cases supported by TCI16093

UC #	Use case objective	Flow Direction	Message Sequence
1	Reset the SUT to the Initial state	TS → SUT	request.SetInitialState

		SUT → TS	response
2	The SUT sets up to send WSM transmissions	TS → SUT SUT -> TS	request.SetWsmTxInfo response
3	The SUT transmits a single or periodic WSMs	SUT → TS TS → SUT	response request.StartWsmTx response
4	The SUT stops transmitting periodic WSMs	TS → SUT SUT → TS	request.StopWsmTx response
5	The SUT starts transmitting WSAs periodically	TS → SUT SUT → TS	request.StartWsaTxPeriodic response
6	The SUT stops transmitting WSAs periodically	TS → SUT SUT → TS	request.StopWsaTxPeriodic response
7	The SUT sets up to receive WSM transmissions via PC5 medium	SUT -> TS TS -> SUT TS -> SUT	request.StartWsmRx response indication
8	The SUT stops receiving WSMs	TS → SUT SUT → TS	request.StopRx response
9	The SUT adds a provider service to WSA	TS → SUT SUT → TS	request.AddWsaProviderService response
10	The SUT changes a provider service to WSA	TS → SUT SUT → TS	request.changeWsaProviderService
11	The SUT deletes a provider service from WSA	TS → SUT SUT → TS	request.DelWsaProviderService response
12	The SUT registers a user service and notifies the TS when it is activated	TS → SUT SUT → TS SUT → TS	request.AddUserService response indication
13	The SUT removes a registered user service	TS → SUT SUT → TS	request.DelUserService response
14	The TS requests IPv6 configuration from the SUT	TS → SUT SUT → TS	request. GetIpv6InterfaceInfo responseInfo
15	The TS requests the SUT to change its IPv6 configuration	TS → SUT SUT → TS	request.SetIpv6Address response
16	The TS starts IPV6 Ping with SUT	TS → SUT SUT → TS	request.startIpv6Ping response
17	The TS stops IPv6 Ping with SUT	TS →SUT SUT → TS	request.stopIpv6Ping response
18	The TS Requests the SUT to send Configuration XML	TS → SUT SUT → TS	Request.sendUeConfigXML response
19	The TS sets configuration of SUT	TS → SUT SUT → TS	request.setUeConfig response indication
20	The TS sets flow configuration of SUT	TS → SUT SUT → TS	request.setFlowConfig response indication
21	The TS sends AT command to SUT	TS → SUT SUT → TS	request.sendATcommand response

22	The TS requests the status of the SUT	TS → SUT SUT → TS	Request.requestSUTStatus response
23	The TS requests Wsm Transmissions Count	TS → SUT SUT → TS	request.requestWsmTxCount response
24	The TS requests WSM Reception Count	TS → SUT SUT → TS	Request.requestWsmRxCount response
25	The TS requests the SUT to reset the Transmission Count	TS → SUT SUT → TS	Request.requestWsmTxCountReset response
26	The TS requests the SUT to reset the Reception Count	TS → SUT SUT → TS	Request.requestWsmRxCountReset

The following dependencies are established among use cases:

- UC1 must precede UC 2, UC4, UC6, UC10, UC12, UC13, UC14
- UC3 must follow UC2 UC5 must follow UC4 UC7 must follow UC6 UC8 must follow UC6
- UC9 must follow UC8
- UC11 must follow UC10
- UC12, UC13, UC14 may follow in any order
- UC15 may occur at any time, including during execution of any other UC.

9.4.2 *Request* messages

Table 31 lists all supported R*equest* messages supported in the *TCI16093* frame. When the SUT sends a *Response* message, it must include the MsgID corresponding to the *Request* message.

Table 31 Listing of *Request* messages

Request Messages	MsgID	Explanation
SetInitialState	1	Request to configure SUT to the Initial state
SetWsmTxInfo	2	Request to set parameters used for transmissions of WSMs
StartWsmTx	3	Request to start transmission of WSMs
StopWsmTx	4	Request to stop transmission of WSMs
StartWsaTxPerdiodic	5	Request to start transmission of WSAs
StopWsaTxPeriodic	6	Request to stop transmission of WSAs
StartWsmRx	7	Request to start receiving WSMs
StopWsmRx	8	Request to stop receiving WSMs
AddWsaProviderService	9	Request to add a service provider to an existing WSA broadcast
ChangeWsaProviderService	10	Request to change a service provider to an existing WSA broadcast
DelWsaProviderService	11	Request to delete a service provider from an existing WSA broadcast
AddUserService	12	Request to add a user service
DelUserService	13	Request to delete a user service
GetIpv6InterfaceInfo	14	Request to SUT to report its IPv6 configuration
SetIpv6Address	15	Request to SUT to set its IPv6 address

StartIpv6Ping	16	Request to SUT to send a ping (ICMP over IPv6)
StopIpv6Ping	17	Request to SUT to stop sending ping (ICMP over IPv6)
sendUeConfigXML	21	Request to SUT to send its own Configuration to the TS
setUeConfig	22	Request to configure specific SUT parameters
setFlowConfig	23	Request to configure the Flow Configuration of the SUT
sendATcommand	24	Request to change AT command parameters of the SUT
requestSutStatus	25	Request to SUT for status
requestWsmTxCount	26	Request to SUT for WSM transmit count
requestWsmRxCount	27	Request to SUT for WSM reception count
requestWsmTxCountReset	28	Request to SUT to reset transmission count
requestWsmRxCountReset	29	Request to SUT to reset reception count

9.3.2.1 SetInitialState
This request is used to set the SUT in initial condition. This request is defined in the *TCI-wsm* module.

9.3.2.2 PC5SetWsmTxInfo
This request is used to configure the SUT's WSM transmission parameters. This request correlates to the pre-defined *SetWsmTxInfo* request in the *TCI-wsm* module.

9.3.2.3 StartWsmTx
This request is used to initiate transmission of WSMs by the SUT. Information about the expected content of the WSM needs to be set via the 8.3.2.2 Dot3SetWsmTxInfo request before. This request correlates to the pre-defined *StartWsmTx* request in the *TCI-wsm* module.

9.3.2.4 StopWsmTx
This request is used to cease transmission of WSMs by the SUT. This request is defined in the *TCI-wsm* module.

9.3.2.5 StartWsaTxPerdiodic
This request is used to initiate transmission of WSA by the SUT. This request is defined in the *TCI-wsm* module.

9.3.2.6 StopWsaTxPeriodic
This request is used to stop the current WSA transmissions by the SUT and delete associated provider services from the *ProviderServiceRequestTable*. This request is defined in the *TCI-wsm* module.

9.3.2.7 PC5StartWsmRx
This request is used to configure the SUT to receive messages and forward corresponding event indications to the TS. This request is defined in the *TCI-wsm* module.

9.3.2.8 StopWsmRx
This request is used to stop the SUT's reception of messages and generation of *indication* messages. This request is defined in the *TCI-wsm* module.

9.3.2.9 AddWsaProviderService
This request is used to add a provider service and update WSA. This request is defined in the *TCI-wsm* module.

9.3.2.10 changeWsaProviderService
This request is used to change a provider service and update WSA. This request is defined in the TCI-wsm module.

9.3.2.11 DelWsaProviderService
This request is used to removes a provider service and updates WSA. This request is defined in the *TCI-wsm* module.

9.3.2.12 AddUserService
This request is used to add a user service to the SUT. This request is defined in the *TCI-wsm* module.

9.3.2.13 DelUserService
This request is used to delete a user service on the SUT previously requested by the *AddUserService* request. This request is defined in the *TCI-wsm* module.

9.3.2.14 GetIpv6InterfaceInfo
This request is used to retrieve IPv6 configuration from the SUT. This request is defined in the *TCI-ip* module.

9.3.2.15 SetIpv6Address
This request is used to set IPv6 address on the SUT. This request is defined in the *TCI-ip* module.

9.3.2.16 StartIpv6Ping
This request is used to request the SUT to transmit a single ping message over IPv6 and receive a ping echo from the remote host. This request is defined in the *TCI-ip* module.

9.3.2.17 StopIpv6Ping
This request is used to stop requesting the SUT to transmit ping messages. This request is defined in the TCI-ip module.

9.3.2.18 SendATcommand
This request is used to send AT commands to the SUT for configuration purposes. This request is defined below:

```
SendATcommand ::= ATcmdInfo (AT+CATM, AT+CCUTLE, etc.)
```

9.3.2.19 SetFlowConfig
This request is used to set the flow configuration of the SUT. This request is defined below:

```
SetFlowConfig ::= SEQUENCE(SIZE(0..32)) OF SEQUENCE {
 flowID                FlowIdentifier
 flowType              ENUMERATED {sps (1), event (2)},
 Pppp                  PPPP,
 serviceId             INTEGER OPTIONAL,
 spsReservationSize    SpsReservationSize OPTIONAL,
 periodicity           TrafficPeriodicity-r14 OPTIONAL,
txPower                P-Max OPTIONAL,
Harq                   BOOLEAN OPTIONAL
txPoolId               INTEGER OPTIONAL,
…
}
```

9.3.2.20 SetUeConfig
This request is used to set the UE configuration of the SUT. This request is defined below:

```
SetUeConfig ::= SEQUENCE {
 earfcn                EARFCN OPTIONAL,
 pMax                    P-Max OPTIIONAL,
 bw                      SL-Bandwidth-r12 OPTIONAL,
 minMcs                  MCS OPTIONAL,
 maxMcs                  MCS OPTIONAL,
 numSubCh                NumSubCh OPTIONAL,
 subChSize               SubChSize OPTIONAL,
 Pdb                     PDB OPTIONAL
…
}
```

9.3.2.21 SendUeConfigXML

This request is used to receive the configuration XML from the SUT. This request is defined below:

```
SendUeConfigXML ::= SEQUENCE {
 counter                 INTEGER (1..127),
 Total                   INTEGER (1..127),
 Crc                     Opaque(SIZE(4)) OPTIONAL,
 Pdu                     Opaque(SIZE(0..tciMtu)),
 …
}
```

9.3.3 *Response* messages

The *Response* message is sent in response to the *Request*. *Response* is defined in the *TCI-CommonTypes* module.

9.3.4 *Indication* messages

The *Indication* message is sent from the SUT to the TS indicating an occurrence of a predefined event. *TCI-16093* defines *Dot3Indication* as follows:

```
Dot3Indication ::= Indication (WITH COMPONENTS{
 radio,
 Event ( e16093PktRx |
         eWsmPktRx |
                  eIpv6PktRx |
                  eIcmp6PktRx |
                  eIpv6ConfigChanged |
                  eDot3ChannelAssigned |
                  eDot3RequestMatchedAvailAppService |
                  eDot2VerificationCompleteWithResult |
                  exception),
 eventParams (WITH COMPONENTS {service} |
              WITH COMPONENTS {wsm} |
              WITH COMPONENTS {ip}
              WITH COMPONENTS {security} |
              )OPTIONAL,
 pdu OPTIONAL,
 exception OPTIONAL
})
```

where *Indication* is defined in the *TCI-indication* module.

Table 32 Indication message

Parameters	**Explanation**

radio	The structure contains radio device (radio0, radio1, etc) and antenna port for transmission of WSAs.
event	Enumerated list of events that when occur, will generate an Indication messages. See 7.3.2 for the list of pre-defined events.
eventParams	Event parameters contain some data related to message reception but not included in the message payload.
pdu	Optional element containing payload of the message identified by the event.
exception	Optional element which is used to report exception. It is included if an exception is reported.

The SUT does not need to send both an *Indication* message with an *exception* parameter and a separate *Exception* message. If the SUT detects an exception, which doesn't not prevent it to receive and process subsequent messages, the SUT must report the exception in the *Indication* message. The SUT must use the *Exception* message if the exception condition causes the SUT to abort generation of *Indication* messages.

Specific details for each type definition are listed in the ASN.1 specification referenced in Appendix A.

9.3.5 *ResponseInfo* messages

This message is used to retrieve configuration information from the SUT.
TCI-16093 defines *Dot3ResponseInfo* as follows:

```
Dot3ResponseInfo ::= ResponseInfo (WITH COMPONENTS{
 msgID,
 resultCode,
 info (WITH COMPONENTS {ipv6InterfaceInfo} |
      WITH COMPONENTS {sutInfo} |
      WITH COMPONENTS {pktCount}) OPTIONAL,
 exception OPTIONAL
})
```

Table 33 ResponseInfo message

Parameters	**Explanation**
msgID	Use the same MsgID from the corresponding *Request* message. MsgIDs are listed in the Table 31.
resultCode	Success or Failure enumerated as 0 or 1 respectively.
info	This parameter contains information requested from the SUT. If SUT detects an error which prevents it to report the requested information, then info parameter is omitted and instead exception parameter is included.
exception	This optional parameter is included if SUT must report exception explaining the possible details of the Failure result code. See details in 8.3.6..

Specific details for each type definition are listed in the ASN.1 specification referenced in Appendix A.

9.3.6 *Exception* messages

Exception is a message sent from the SUT to the TS. It is used to report exception conditions to the TS. *Exception* and defined in the *TCI-CommonTypes* module.

9.4 TCI29451 frame

Use cases supported by TCI29451 are listed in Table 41.

Table 41 Use cases supported by TCI29451

UC #	Request/Response Messages	Flow Direction	Message Sequence
1	Set the SUT to the Initial state	TS → SUT SUT → TS	request.SetInitialState response
2	The SUT transmits periodic BSMs	TS → SUT SUT → TS TS → SUT SUT → TS	request.StartBsmTx response
3	The SUT stops transmitting periodic BSMs	TS → SUT SUT → TS	request.StopBsmTx response
4	Turn congestion mitigation of the SUT on or off	TS → SUT SUT → TS	request.EnableCongestionMitigation response
5	Set the Temporary ID of the SUT	TS → SUT SUT → TS	request.SetTemporaryId response
6	Set the transmission of the SUT	TS → SUT SUT → TS	request.SetTransmissionState response
7	Set the SteeringWheelAngle of the SUT	TS → SUT SUT → TS	request.setSteeringWheelAngle response
8	Set the Brake System Status of the SUT	TS → SUT SUT → TS	request.setBrakeSystemStatus response
9	Set the vehicle size of the SUT	TS → SUT SUT → TS	request.setVehicleSize response
10	Set the exterior lights status of the SUT	TS → SUT SUT → TS	request.SetExteriorLightsStatus response
11	Set the vehicle event flags of the SUT	TS → SUT SUT → TS	Request.SetVehicleEventFlags response
12	The SUT starts receiving BSMs	TS → SUT SUT → TS	request.StartBsmRx response
13	The SUT stops receiving BSMs	TS → SUT SUT → TS	request.StopBsmRX response

9.4.1 Request messages

Table 41 lists all supported *request* messages. When the SUT sends a *response* message, it must include the *MsgID* corresponding to the *request* message.

Table 41 Request supported in TCI29451 frame

Request Messages	MsgID	Explanation
SetInitialState	1	Set the SUT to the Initial state
StartBsmTx	2	Begin transmission of BSMs

StopBsmTx	3	Stop transmission of BSMs
StartBsmRx	4	Begin reception of BSMs
StopBsmRx	5	Stop reception of BSMs
EnableCongestionMitigation	6	Enable or disable the congestion mitigation on the SUT
SetTemporaryId	10	Set the temporary ID of the SUT, overwriting the current ID
SetTransmissionState	11	Set the transmission state of the SUT, overwriting its current transmission
SetSteeringWheelAngle	12	Set the steering wheel angle of the SUT, overwriting the current steering wheel angle
SetBrakeSystemStatus	13	Enable or disable the brake pedal status of the SUT
SetVehicleSize	14	Set the Vehicle Size in the SUT
SetExteriorLights	15	Set the exterior lights status of the SUT, overwriting its current light status
SetVehicleEventFlags	16	Set the vehicle flags of the SUT, overwriting its current flags

9.4.1.1 SetInitialState

This request is used to set the SUT in initial condition. The initial condition defines the initial state in which the SUT must be to carry out each test case.

9.4.1.2 EnableGpsInput (Deprecated)

This request was moved to the *TCI-SutControl* module. Refer to 8.5.2.6 enableGpsInputEnableGpsInput.

9.4.1.3 StartBsmTx

This request is used to start BSM transmission on the SUT. The *StartWSMTx* request is predefined in *TCI-wsm* module.

```
StartBsmTx ::= StartWsmTx (WITH COMPONENTS {
    psid  (WITH COMPONENTS {content (32)}),
    radio,
    repeatRate ABSENT,
    payload ABSENT
})
```

9.4.1.4 StopBsmTx

This request is used to stop BSM transmission on the SUT. The *StopWSMTx* request is predefined in *TCI-wsm* module.

```
StopBsmTx ::= StopWsmTx (WITH COMPONENTS {
    psid (WITH COMPONENTS {content  (32)})
})
```

9.4.1.5 EnableCongestionMitigation

This request sets the congestion mitigation of the SUT.

```
EnableCongestionMitigation ::= BOOLEAN
```

9.4.1.6 SetTemporaryId

This request sets the temporary ID of the SUT. The definition of data units is adopted from [10].

```
SetTemporaryId ::= OCTET STRING (SIZE(4))
```

9.4.1.7 SetLatitude (Deprecated)

This request was moved to the *TCI-SutControl* module. Refer to 8.5.2.7 setLatitudeSetLatitude.

9.4.1.8 SetLongitude (Deprecated)

This request was moved to the *TCI-SutControl* module. Refer to 8.5.2.8 setLongitudeSetLongitude.

9.4.1.9 SetElevation (Deprecated)

This request was moved to the *TCI-SutControl* module. Refer to 8.5.2.9 setElevationSetElevation.

9.4.1.10 SetPositionalAccuracy (Deprecated)

This request was moved to the *TCI-SutControl* module. Refer to 8.5.2.10 setPositionalAccuracySetPositionalAccuracy

9.4.1.11 SetTransmissionState

This request is used to set the vehicle transmission state of the SUT.

```
SetTransmissionState ::= ENUMERATED {
    neutral         (0),
    park            (1),
    forwardGears    (2),
    reverseGears    (3),
    reserved1       (4),
    reserved2       (5),
    reserved3       (6),
    unavailable     (7)
}
```

Parameters	**Explanation**
neutral	The vehicle is set to neutral gear
park	The vehicle is set to park
forwardGears	The vehicle is set to forward gear
reverseGears	The vehicle is set to reverse gear
reserved1	Reserved for additional gears
reserved2	Reserved for additional gears
reserved3	Reserved for additional gears
unavailable	Vehicle transmission is set to unavailable

9.4.1.12 SetSpeed (Deprecated)

This request was moved to the *TCI-SutControl* module. Refer to 8.5.2.11 setSpeedSetSpeed.

9.4.1.13 SetHeading (Deprecated)

This request was moved to the *TCI-SutControl* module. Refer to 8.5.2.12 setHeadingSetHeading.

9.4.1.14 SetSteeringWheelAngle

The request is used to set the steering wheel angle of the SUT.

```
SetSteeringWheelAngle ::= INTEGER (-126... 127)
```

9.4.1.15 SetAccelerationSet4Way (Deprecated)

This request was moved to the *TCI-SutControl* module. Refer to 8.5.2.13 setAccelerationset4WaySetAccelerationSet4Way.

9.4.1.16 SetBrakeSystemStatus

The request is used to set the status of the brake system of the SUT.

```
SetBrakeSystemStatus ::= SEQUENCE {
    brakeAppliedStatus       BIT STRING {
        unavailable (0),
        leftFront   (1),
        leftRear    (2),
        rightFront  (3),
        rightRear   (4)
    },
    tractionControlStatus    ENUMERATED {
        unavailable (0),
        off         (1),
        on          (2),
        engaged     (3)
    },
    antiLockBrakeStatus      ENUMERATED {
        unavailable (0),
        off         (1),
        on          (2),
        engaged     (3)
    },
    stabilityControlStatus   ENUMERATED {
        unavailable (0),
        off         (1),
        on          (2),
        engaged     (3)
    },
    brakeBoostApplied        ENUMERATED {
        unavailable (0),
        off         (1),
        on          (2)
    },
    auxiliaryBrakeStatus     ENUMERATED {
        unavailable (0),
        off         (1),
        on          (2),
        reserved    (3)
    }
}
```

9.4.1.17 SetVehicleSize

The request sets the vehicle size in the SUT.

```
SetVehicleSize ::= SEQUENCE{
    vehicleWidth     INTEGER(0 .. 1023),
    vehicleLength    INTEGER(0 .. 4095)
}
```

9.4.1.18 SetExteriorLights

The request sets the exterior lights in the SUT.

```
SetExteriorLights ::= BIT STRING
{
   lowBeamHeadlightsOn       (0),
   highBeamHeadlightsOn      (1),
   leftTurnSignalOn          (2),
   rightTurnSignalOn         (3),
   hazardSignalOn            (4),
   automaticLightControlOn   (5),
   daytimeRunningLightsOn    (6),
   fogLightOn                (7),
   parkingLightsOn           (8)
}
```

9.4.1.19 SetVehicleEventFlags

```
SetVehicleEventFlags ::= BIT STRING {
    eventHazardLights               (0),
    eventStopLineViolation          (1),
    eventABSactivated               (2),
    eventTractionControlLoss        (3),
    eventStabilityControlActivated  (4),
    eventHazardousMaterials         (5),
    eventReserved1                  (6),
    eventHardBraking                (7),
    eventLightsChanged              (8),
    eventWipersChanged              (9),
    eventFlatTire                   (10),
    eventDisabledVehicle            (11),
    eventAirBagDeployment           (12)
}
```

9.4.1.20 StartBsmRx

This request starts BSM reception on the SUT. The *StartWsmRx* request is predefined in *TCI-wsm* module

```
StartBsmRx ::= StartWsmRx (WITH COMPONENTS {
    psid   (WITH COMPONENTS {content    (32)}),
    radio ( WITH COMPONENTS { ..., antenna ABSENT }),
    channelIdentifier (172),
    timeSlot (continuous),
    eventHandling
    })
```

9.4.1.21 StopBsmRx

This request starts BSM reception on the SUT. The *StopWsmRx* request is predefined in *TCI-wsm* module

```
StopBsmRx ::= StopWsmRx (WITH COMPONENTS {
        psid (WITH COMPONENTS {content  (32)})
    })
```

9.4.1.22 setGpsTime (Deprecated)

This request was moved to the *TCI-SutControl* module. Refer to 8.5.2.14 setGpsTimeSetGpsTime.

9.4.2 *Response* messages

The *Response* message is sent in response to the *Request*. *Response* is defined in the *TCI-CommonTypes* module.

9.4.3 *Indication* messages

The *Indication* message is sent from the SUT to the TS indicating an occurrence of a predefined event. *TCI-29451* defines *D2945Indication* as follows:

```
D2945Indication ::= Indication (WITH COMPONENTS {
    radio,
    event ( eWsmPktRx |
           exception),
    eventParams (WITH COMPONENTS {wsm}
                ) OPTIONAL,
    pdu OPTIONAL,
    exception OPTIONAL
 })
```

where *Indication* is defined in the *TCI-indication* module.

9.4.4 *ResponseInfo* messages

TCI-29451 does not use *ResponseInfo* messages.

9.4.5 *Exception* messages

Exception is a message sent from the SUT to TS. It is used to report exception conditions to the TS. *Exception* is defined in the *TCI-CommonTypes* module.

9.5 TCI31611 frame

9.5.1 Supported use cases

Use cases (UC) supported by TCI16093PC5 are listed in Table 40.

Note, the use cases correlate to the previous Supported Use Cases section J2945/1.

Table 40 Use cases supported by TCI31611

1	Set the SUT to the Initial state	TS → SUT SUT → TS	request.SetInitialState response
2	The SUT transmits periodic BSMs	TS → SUT SUT → TS TS → SUT SUT → TS	request.StartBsmTx response
3	The SUT stops transmitting periodic BSMs	TS → SUT SUT → TS	request.StopBsmTx response
4	Turn congestion mitigation of the SUT on or off	TS → SUT SUT → TS	request.EnableCongestionMitigation response
5	Set the Temporary ID of the SUT	TS → SUT SUT → TS	request.SetTemporaryId response
6	Set the transmission of the SUT	TS → SUT SUT → TS	request.SetTransmissionState response
7	Set the SteeringWheelAngle of the SUT	TS → SUT SUT → TS	request.setSteeringWheelAngle response
8	Set the Brake System Status of the SUT	TS → SUT SUT → TS	request.setBrakeSystemStatus response
9	Set the vehicle size of the SUT	TS → SUT SUT → TS	request.setVehicleSize response
10	Set the exterior lights status of the SUT	TS → SUT SUT → TS	request.SetExteriorLightsStatus response
11	Set the vehicle event flags of the SUT	TS → SUT SUT → TS	Request.SetVehicleEventFlags response
12	The SUT starts receiving BSMs	TS → SUT SUT → TS	request.StartBsmRx response

13	The SUT stops receiving BSMs	TS → SUT SUT → TS	request.StopBsmRX response

The following dependencies are established among use cases:

- UC1 must precede UC 2, UC4, UC6, UC10, UC12, UC13, UC14
- UC3 must follow UC2 UC5 must follow UC4 UC7 must follow UC6 UC8 must follow UC6
- UC9 must follow UC8
- UC11 must follow UC10
- UC12, UC13, UC14 may follow in any order
- UC15 may occur at any time, including during execution of any other UC.

9.5.2 *Request* messages

Table 31 lists all supported R*equest* messages supported in the *TCI16093* frame. When the SUT sends a *Response* message, it must include the MsgID corresponding to the *Request* message.

Table 31 Listing of *Request* messages

Request Messages	MsgID	Explanation
SetInitialState	1	Set the SUT to the Initial state
StartBsmTx	2	Begin transmission of BSMs
StopBsmTx	3	Stop transmission of BSMs
StartBsmRx	4	Begin reception of BSMs
StopBsmRx	5	Stop reception of BSMs
EnableCongestionMitigation	6	Enable or disable the congestion mitigation on the SUT
SetTemporaryId	10	Set the temporary ID of the SUT, overwriting the current ID
SetTransmissionState	11	Set the transmission state of the SUT, overwriting its current transmission
SetSteeringWheelAngle	12	Set the steering wheel angle of the SUT, overwriting the current steering wheel angle
SetBrakeSystemStatus	13	Enable or disable the brake pedal status of the SUT
SetVehicleSize	14	Set the Vehicle Size in the SUT
SetExteriorLights	15	Set the exterior lights status of the SUT, overwriting its current light status
SetVehicleEventFlags	16	Set the vehicle flags of the SUT, overwriting its current flags

9.5.2.1 SetInitialState

This request is used to set the SUT in initial condition. The initial condition defines the initial state in which the SUT must be to carry out each test case.

9.5.2.2 StartBsmTx

This request is used to start BSM transmission on the SUT. The *StartWSMTx* request is predefined in *TCI-wsm* module.

```
StartBsmTx ::= StartWsmTx (WITH COMPONENTS {
    psid  (WITH COMPONENTS {content (32)}),
    radio,
    repeatRate ABSENT,
    payload ABSENT
})
```

9.5.2.3 *StopBsmTx*

This request is used to stop BSM transmission on the SUT. The *StopWSMTx* request is predefined in *TCI-wsm* module.

```
StopBsmTx ::= StopWsmTx (WITH COMPONENTS {
    psid (WITH COMPONENTS {content  (32)})
})
```

9.5.2.4 *EnableCongestionMitigation*

This request sets the congestion mitigation of the SUT.

```
EnableCongestionMitigation ::= BOOLEAN
```

9.5.2.5 *SetTemporaryId*

This request sets the temporary ID of the SUT. The definition of data units is adopted from [10].

```
SetTemporaryId ::= OCTET STRING (SIZE(4))
```

9.5.2.6 *SetTransmissionState*

This request is used to set the vehicle transmission state of the SUT.

```
SetTransmissionState ::= ENUMERATED {
    neutral         (0),
    park            (1),
    forwardGears    (2),
    reverseGears    (3),
    reserved1       (4),
    reserved2       (5),
    reserved3       (6),
    unavailable     (7)
}
```

Parameters	**Explanation**
neutral	The vehicle is set to neutral gear
park	The vehicle is set to park
forwardGears	The vehicle is set to forward gear
reverseGears	The vehicle is set to reverse gear
reserved1	Reserved for additional gears
reserved2	Reserved for additional gears
reserved3	Reserved for additional gears
unavailable	Vehicle transmission is set to unavailable

9.5.2.7 *SetSteeringWheelAngle*

The request is used to set the steering wheel angle of the SUT.

```
SetSteeringWheelAngle ::= INTEGER (-126... 127)
```

9.5.2.8 *SetBrakeSystemStatus*

The request is used to set the status of the brake system of the SUT.

```
SetBrakeSystemStatus ::= SEQUENCE {
    brakeAppliedStatus       BIT STRING {
        unavailable (0),
        leftFront   (1),
        leftRear    (2),
        rightFront  (3),
        rightRear   (4)
    },
    tractionControlStatus    ENUMERATED {
        unavailable (0),
        off         (1),
        on          (2),
        engaged     (3)
    },
    antiLockBrakeStatus      ENUMERATED {
        unavailable (0),
        off         (1),
        on          (2),
        engaged     (3)
    },
    stabilityControlStatus   ENUMERATED {
        unavailable (0),
        off         (1),
        on          (2),
        engaged     (3)
    },
    brakeBoostApplied        ENUMERATED {
        unavailable (0),
        off         (1),
        on          (2)
    },
    auxiliaryBrakeStatus     ENUMERATED {
        unavailable (0),
        off         (1),
        on          (2),
        reserved    (3)
    }
}
```

9.5.2.9 SetVehicleSize

The request sets the vehicle size in the SUT.

```
SetVehicleSize ::= SEQUENCE{
    vehicleWidth     INTEGER(0 .. 1023),
    vehicleLength    INTEGER(0 .. 4095)
}
```

9.5.2.10 SetExteriorLights

The request sets the exterior lights in the SUT.

```
SetExteriorLights ::= BIT STRING
{
   lowBeamHeadlightsOn       (0),
   highBeamHeadlightsOn      (1),
   leftTurnSignalOn          (2),
   rightTurnSignalOn         (3),
   hazardSignalOn            (4),
   automaticLightControlOn   (5),
   daytimeRunningLightsOn    (6),
   fogLightOn                (7),
   parkingLightsOn           (8)
}
```

9.5.2.11 SetVehicleEventFlags

```
SetVehicleEventFlags ::= BIT STRING {
    eventHazardLights               (0),
    eventStopLineViolation          (1),
    eventABSactivated               (2),
    eventTractionControlLoss        (3),
    eventStabilityControlActivated  (4),
    eventHazardousMaterials         (5),
    eventReserved1                  (6),
    eventHardBraking                (7),
    eventLightsChanged              (8),
    eventWipersChanged              (9),
    eventFlatTire                   (10),
    eventDisabledVehicle            (11),
    eventAirBagDeployment           (12)
}
```

9.5.2.12 StartBsmRx

This request starts BSM reception on the SUT. The *StartWsmRx* request is predefined in *TCI-wsm* module

```
StartBsmRx ::= StartWsmRx (WITH COMPONENTS {
    psid   (WITH COMPONENTS {content    (32)}),
    radio ( WITH COMPONENTS { ..., antenna ABSENT }),
    channelIdentifier ABSENT,
    timeSlot ABSENT,
    eventHandling
    })
```

9.5.2.13 StopBsmRx

This request starts BSM reception on the SUT. The *StopWsmRx* request is predefined in *TCI-wsm* module

```
StopBsmRx ::= StopWsmRx (WITH COMPONENTS {
        psid (WITH COMPONENTS {content  (32)})
    })
```

9.4.2 *Response* messages

The *Response* message is sent in response to the *Request*. *Response* is defined in the *TCI-CommonTypes* module.

9.4.3 *Indication* messages

The *Indication* message is sent from the SUT to the TS indicating an occurrence of a predefined event. *TCI-29451* defines *D2945Indication* as follows:

```
D31611Indication ::= Indication (WITH COMPONENTS
{
    radio,
    event ( eWsmPktRx |
           exception),
    eventParams (WITH COMPONENTS {wsm}
                 ) OPTIONAL,
    pdu OPTIONAL,
    exception OPTIONAL
 })
```

where *Indication* is defined in the *TCI-indication* module.

9.4.4 *ResponseInfo* messages

TCI-29451 does not use *ResponseInfo* messages.

9.4.5 *Exception* messages

Exception is a message sent from the SUT to TS. It is used to report exception conditions to the TS. *Exception* is defined in the *TCI-CommonTypes* module.

9.6 TCISutControl frame

9.6.1 Supported use cases

Use cases (UC) supported by *TCI-SutControl* are listed in Table 34.

Table 34 Use cases supported by TCI16093

UC #	Use case objective	Flow Direction	Message Sequence
1	Request the SUT to shut down.	TS -> SUT SUT -> TS	request.Shutdown response
2	Request the SUT to restart.	TS -> SUT SUT -> TS	request.Restart response
3	Request SUT status to accept new commands.	TS -> SUT SUT -> TS	request.RequestSutAvailability response
4	Request SUT version information	TS → SUT SUT → TS	request.RequestSutInfo responseInfo
5	Provide information about Test ID to the SUT	TS → SUT SUT → TS	request.SutStatus response
6	TS Sets Test ID	TS → SUT	request.setTestId
7	Request enable/disable GPS input of the SUT	TS → SUT SUT → TS	request.enableGpsInput response
8	Set the latitude of the SUT	TS → SUT SUT → TS	request.setLatitude response
9	Set the longitude of the SUT	TS → SUT SUT → TS	request.setLongitude response
10	Set the elevation of the SUT	TS → SUT SUT → TS	request.setElevation response
11	Set the positional accuracy of the SUT	TS → SUT SUT → TS	request.setPositionalAccuracy response
12	Set the speed of the SUT	TS → SUT SUT → TS	request.setSpeed response
13	Set the heading of the SUT	TS → SUT SUT → TS	request.setHeading response
14	Set the 4-way acceleration of the SUT	TS → SUT SUT → TS	request.setAccelerationSet4Way response

15	Set the GPS time of the SUT	TS → SUT SUT → TS	request.setGpsTime response

9.6.2 *Request* messages

Table 35 lists all supported R*equest* messages in the *TCI-SutControl* module.

Table 35 Listing of *Request* messages

Request Messages	MsgID	Explanation
Shutdown	1	Request to shut the SUT down.
Restart	2	Request to restart the SUT.
RequestSutAvailability	3	Request SUT availability status.
RequestSutInfo	4	Request information about SUT version
SetTestId	5	Send Test ID information to the SUT
EnableGpsInput	6	Enable/Disable GPS on the SUT
SetLatitude	7	Set the latitude of the SUT
SetLongitude	8	Set the longitude of the SUT
SetElevation	9	Set the elevation of the SUT
SetPositionalAccuracy	10	Set the positional accuracy of the SUT
SetSpeed	11	Set the speed of the SUT
SetHeading	12	Set the heading of the SUT
SetAccelerationSet4Way	13	Set the 4-way acceleration of the SUT
SetGpsTime	14	Set the GPS time of the SUT
RequestSutStatus	15	Request the Status of the SUT

9.6.2.1 Shutdown

This request is used to command the SUT to shut down and power off. If complete power off is not supported, the device must enter a state where the CPU is halted, and power draw is minimized.

9.6.2.2 Restart

This request is used to command the SUT to restart. The "restart" is meant to be interpreted as it is used in defining certain requirements in SAE J2945/1 [9]. Therefore, this request must trigger the device to perform certain activities which must occur upon the device restart, i.e., change security certificates, change MAC address to a new random value, etc.

9.6.2.3 RequestSutAvailability

This request is used to poll the availability of the SUT after a preceding restart. If the SUT is ready to receive commands from the TS, it responds back to the TS with a Response message and ResultCode = rcSuccess. The SUT is not ready if it does not respond within the response timeout of **50ms** or includes the ResultCode = rcFailure.

9.6.2.4 RequestSutInfo

This request is used to obtain version information from the SUT. This request must be answered with the *SutResponseInfo* message. The version information can be referenced in test reports and other test documentation.

9.6.2.5 SetTestId

This request is used to send a Test identifier to the SUT. The Test ID is a text string e.g., "TP-16093-WSMMST-BV-01" which the SUT can reference in its own log file. This message could be used for identifying tests in all TCI frames, i.e., TCI16093, TCI80211, TCI16094, etc.

There is no time restriction when the TS can send this message. Though, it is recommended that the *SetTestId* message is sent at the beginning of each individual test, after the *request.SetInitialState* --> *response* sequence is completed.

9.6.2.6 EnableGpsInput

This request sets GPS Input to true or false. If it is set to *True*, the SUT will use its own GPS data input to retrieve positioning data. If it is set to *False*, the SUT will retrieve all positioning related data via TCI messages SetLatitude, SetLongitude, SetElevation, SetPositionalAccuracy, SetSpeed, SetHeading, SetAccelerationSet4Way, and SetGpsTime.

Note that in case not all data were provided via TCI, the SUT should use previously stored values or initial values.

```
EnableGpsInput ::= BOOLEAN
```

9.6.2.7 SetLatitude

This request sets the latitude of the position of the SUT.

Note that this command can only be processed if the GPS data input was set to *False* via the *EnableGpsInput* request.

```
SetLatitude ::= Latitude
```

9.6.2.8 SetLongitude

This request sets the longitude of the position of the SUT.

Note that this command can only be processed if the GPS data input was set to *False* via the *EnableGpsInput* request.

```
SetLongitude ::= Longitude
```

9.6.2.9 SetElevation

This request sets the elevation of the position of the SUT.

Note that this command can only be processed if the GPS data input was set to *False* via the *EnableGpsInput* request.

```
 SetElevation ::= Elevation
```

9.6.2.10 SetPositionalAccuracy

This request sets the positional accuracy of the position of the SUT.

Note that this command can only be processed if the GPS data input was set to *False* via the *EnableGpsInput* request.

```
  SetPositionalAccuracy ::= SEQUENCE{
    semiMajorAxisAccuracy   INTEGER (0 .. 255),
    semiMinorAxisAccuracy   INTEGER (0 .. 255),
    semiMajorAxisOrientation   INTEGER (0 .. 65535)
}
```

9.6.2.11 SetSpeed

This request sets the current speed of the SUT.

Note that this command can only be processed if the GPS data input was set to *False* via the *EnableGpsInput* request.

```
SetSpeed ::= INTEGER (0 .. 8191)
```

9.6.2.12 SetHeading

This request sets the heading of the SUT.

Note that this command can only be processed if the GPS data input was set to *False* via the *EnableGpsInput* request.

```
SetHeading ::= INTEGER (0 .. 28800)
```

9.6.2.13 SetAccelerationSet4Way

This request sets the 4-way acceleration of the SUT.

Note that this command can only be processed if the GPS data input was set to *False* via the *EnableGpsInput* request.

```
SetAccelerationSet4Way ::= SEQUENCE {
    longAcceleration INTEGER (-2000 .. 2001),
    latAcceleration INTEGER (-2000 .. 2001),
    verticalAcceleration INTEGER (-127 .. 127),
    yawRate INTEGER (-32767 .. 32767)
}
```

9.6.2.14 SetGpsTime

This request sets the GPS time of the SUT.

Note that this command can only be processed if the GPS data input was set to *False* via the *EnableGpsInput* request.

```
SetGpsTime ::= Time64
```

9.6.3 Response messages

The *Response* message is sent in response to the *Request*. *Response* is defined in the *TCI-CommonTypes* module.

9.6.4 *ResponseInfo* messages

This message is used to retrieve version information from the SUT.
TCI-SutControl defines *SutResponseInfo* as follows:

```
SutResponseInfo ::= ResponseInfo (WITH COMPONENTS {
   msgID,
   resultCode,
   info (WITH COMPONENTS {sutInfo}) OPTIONAL,
   exception OPTIONAL
   })
```

Table 36 ResponseInfo message

Parameters	**Explanation**

msgID	Use the same MsgID from the corresponding *Request* message. MsgIDs are listed in the Table 35Table 31.
resultCode	Success or Failure enumerated as 0 or 1, respectively.
info	This parameter contains information requested from the SUT. If SUT detects an error which prevents it to report the requested information, then info parameter is omitted and instead the exception parameter is included.
exception	This optional parameter is included if SUT must report exception explaining the possible details of the Failure result code. See details in 8.5.5

The *SutResponseInfo* is derived from the *ResponseInfo* definition in the *TCI-responseInfo* module. Specific details for each type definition are listed in the ASN.1 specification referenced in Appendix A.

9.6.5 *Exception* messages

Exception is a message sent from the SUT to the TS. It is used to report exception conditions to the TS. *Exception* is defined in the *TCI-CommonTypes* module.

Appendix A: TCI protocol ASN.1 definition

This appendix contains listing of all data types defined in the ASN.1 for the TCI protocol. Data types are listed under the corresponding module name where they are defined.

The current TCI protocol ASN.1 definition file is posted in github at the following location:

https://github.com/OmniAirConsortium/TCIV3/tree/TCIv3/ASN1

```
TCIdispatch.asn
 TCIMsg
 Frame

TCI16093DSRC.asn
 TCI16093DSRC
 Request
 MESSAGE-ID-AND-TYPE
 Dot3SetWsmTxInfo
 Dot3StartWsmTx
 Dot3StartWsmRx
 Dot3StartWsaTxPerdiodic
 Dot3Indication
 Dot3ResponseInfo

TCI16093PC5.asn
TCI16093PC5
 Request
 MESSAGE-ID-AND-TYPE
 SendATcommand
 PC5SetWsmTxInfo
 PC5StartWsaTxPerdiodic
 SetFlowConfig
 SetUeConfig
 SendUeConfigXML
 StartWsmTx
 StartWsmRx
 Pc5AddUserService
 Pc5Indication
 Pc5ResponseInfo
```

TCI16094.asn

```
 TCI16094
 Request
 MESSAGE-ID-AND-TYPE
 Dot4SetWsmTxInfo
 Dot4StartWsmTx
 Dot4Indication
 Dot4ResponseInfo
```

TCI29451.asn

```
 TCI29451
 Request
 MESSAGE-ID-AND-TYPE
 StartBsmTx
 StopBsmTx
 StartBsmRx
 StopBsmRx
 EnableCongestionMitigation
 SetTemporaryID
 SetTransmissionState
 SetSteeringWheelAngle
 SetBrakeSystemStatus
 SetVehicleSize
 SetVehcileEventFlags
 SetExteriorLights
 D2945Indication
```

TCI31611.asn

```
 TCI31611
 Request
 MESSAGE-ID-AND-TYPE
 StartBsmTx
 StopBsmTx
 StartBsmRx
 StopBsmRx
 EnableCongestionMitigation
 SetTemporaryID
 SetTransmissionState
 SetSteeringWheelAngle
 SetBrakeSystemStatus
 SetVehicleSize
 SetVehcileEventFlags
 SetExteriorLights
 D31611Indication
```

TCI80211.asn

```
 TCI80211
 Request
 MESSAGE-ID-AND-TYPE
 Dot11SetWsmTxInfo
 Dot11StartWsmTx
 Dot11Indication
```

TCIEventHandling.asn
```
 EventHandling
 RxFlag
 EventFlag
 SecurityFlag
 EventParamsChoice
```

TCIindication.asn
```
 Indication
 Event
 EventParams
 Pdu
 ServiceParameters
 WsmParameters
 IpParameters
 RadioParameters
 SecResultParams
```

TCIip.asn
```
 AddTxProfile
 DelTxProfile
 GetIPv6InterfaceInfo
 SetIPv6Address
 IPv6TxRecord
 StartIPv6Tx
 StopIPv6Tx
 StartIPv6Ping
 StopIPv6Ping
 IPv6RxRecord
 StartIPv6Rx
 StopIPv6Rx
```

TCIresponseInfo.asn
```
 ResponseInfo
 InfoContent
 Ipv6InterfaceInfo
 SutInfo
 SutStatus
 VersionInfoBlock
 ATcmdInfo
 PacketCount
```

TCISutControl.asn
```
 TCISutControl
 MESSAGE-ID-AND-TYPE
 Request
 MessageTypes
 SetTestId
 Shutdown
 Restart
 RequestSuutAvailability
 RequestSutInfo
 RequestSutStatus
 SutResponseInfo
 EnableGpsInput
 SetGpsTime
 SetLatitude
 SetLongitude
 SetElevation
 SetPositionalAccuracy
 SetSpeed
 SetHeading
 SetAccelerationSet4Way
```

`TCIwsm.asn`

```
setInitialState
SetWsmTxInfo
StartWsmTx
StopWsmTx
StartWsmRx
StopWsmRx
StartWsmTxPerdiodic
StopWsaTxPeriodic
AddWsaProviderService
ChangeWsaProviderService
delWsaProviderService
AddUserService
DelUserService
RequestWsmTxCount
RequestWsmTxCountReset
RequestWsmRxCount
RequestWsmRxCountReset
ContentType
SignerIdentifierType
SecurityPermission
SecurityContext
WaveElementsIncluded
UserRequestType
WsaType
ServiceInfo
ChannelOptions
```

`TCICommonTypes.asn`

```
PduData
PduType
dsrcMtu
ltev2xMtu
ipMtu
tciMtu
Opaque
HashedId8
Response
ResultCode
Exception
ExceptionType
Module
ExceptionId
ExceptionText
RadioInterface
Radio
Antenna
Timeslot
RCPI
UserPriority
Time64
```

```
Psid
RepeatRate
MsgID
```

`WEE.ASN and WSA.ASN are imported from ASN.1 for IEEE 1609.3V3D6`

`wee.asn`

```
EXT-TYPE
Extension
IPv6Address
MACaddress
TXpower80211
ChannelNumber80211
```

`WSA.asn is modified to import VarLengthNumber from TCI-CommonTypes`

`wsa.asn`

```
AdvertiserIdentifier
ProviderServiceContext
ServiceInfoExts
ChannelInfos
RoutingAdvertisement
```

Appendix B: TCIProxyCv2X

Appendix B of this document consist of guidance on how to implement the new TCI proxy for LTE V2X devices.

General:

TCIProxyCv2X is a software running on a device with a Cv2X radio. It will handle all conversion of messages between two interfaces. (UDP message exchange on the ethernet interface and the wireless CV2X frames). The new TCI Proxy Code is present on the OmniAir Consortium github at the link below:

https://github.com/OmniAirConsortium/TCIV3/blob/TCIv3/ASN1/TCIproxyCv2x.asn

Proxy Setup:

There are two use cases for the Proxy device. Both are presented in table B-1.

Table B-1

Proxy Setup	Setup objectives	Flow Direction
1	Receive message from the TS	TS → SUT
2	Receive frames containing WSM/WSA messages from a SUT	SUT → TS

Supported use cases

Use cases (UC) supported by *TCI-SutControl* are listed in Table 34.

Table B-2 Use cases

UC #	Use case objective	Flow Direction	Message Sequence

1	Set Initial State.	TS -> PXY PXY -> TS	Request.setInitialState response
2	SendUeConfigXML	TS -> PXY PXY -> SUT SUT -> PXY	request.sendUeConfigXML response response
3	SetFlowConfig	TS -> SUT SUT -> TS	request.setFlowConfig response
4	SetUeConfig	TS → SUT SUT → TS	request.SetUeConfig responseInfo
5	Send AT Command	TS → SUT SUT → TS	request.sendATcommand response
6	StartUDPProxyTx	TS → SUT	Request.startUdpProxyTx
7	StartUdpProxyRx	TS → SUT SUT → TS	request.startUdpProxyRx response
8	StopUdpProxy	TS → SUT SUT → TS	request.stopUdpProxy response

Request messages

Table 35 lists all supported R*equest* messages in the *TCI-SutControl* module.

Table B-3 Listing of *Request* messages

Request Messages	MsgID
SetInitialState	1
sendUeConfigXML	2
setFlowConfig	3
setUeConfig	4
SendATcommand	5
startUdpProxyTx	6
StartUdpProxyRx	7
StopUdpProxy	8

StartUdpProxyTx

This request is used to start proxy transmission of packets. It is shown below:

```
StartUdpProxyTx ::= SEQUENCE {
 port                    IpPort,
 radio                   RadioInterface,
 flowId                  flowIdentifier OPTIONAL,
 pc5Mtu                  INTEGER(0..65535) DEFAULT ltev2xMtu,
…
}
```

StartUdpProxyRx

This request is used to start proxy reception of packets. It is shown below:

```
StartUdpProxyRx ::= SEQUENCE {
 destAddress            IpAddress,
 destPort               IpPort,
 Radio                  RadioInterface,
 flowID                 FlowIdentifier OPTIONAL
 Pc5Mtu                 INTEGER(0..65535) DEFAULT ltev2xMtu,
 proto                  ENUMERATED {
                           pacp (0) },
 options                BIT STRING {
                             radiotap (0)} OPTIONAL,
…
}
```

StopUdpProxy

This request is used to stop proxy transmission/reception of packets. It is shown below:

```
StartUdpProxyTx ::= SEQUENCE {
 port                   IpPort,
 radio                  RadioInterface,
 flowId                 flowIdentifier OPTIONAL,
 pc5Mtu                 INTEGER(0..65535) DEFAULT ltev2xMtu,
…
}
```

PC5ProxyResponseInfo

This message is used to retrieve configuration information from the SUT. It is defined in the *TCI-responseInfo* module. A *ResponseInfo* message must be triggered within **50ms** after an SUT received a *Request* message. If no *ResponseInfo* is received, the TS will attempt to re-initialize the SUT or may request user assistance.

```
PC5ProxyResponseInfo ::= ResponseInfo (WITH COMPONENTS {
 msgID,
 resultCode,
 info (
   WITH COMPONENTS {atCmdInfo} |
   WITH COMPONENTS {pktCount}  |
   WITH COMPONENTS {sutStatus} ) OPTIONAL,
 Exception OPTIONAL
}
```

IpAddress

This request sets the IPAddress of the proxy

```
IpAddress ::= UTF8String(SIZE(2..255))
```

IpPort

This request sets the IpPort of the Proxy

```
IpPort ::= INTEGER(0..65535)
```

Open Issues

`TCIwsm.asn → ServiceInfos commented out`

`MessageType section in separate modules need to be more consistent`

Perdiodic or Periodic??

◙ End of Document ◙