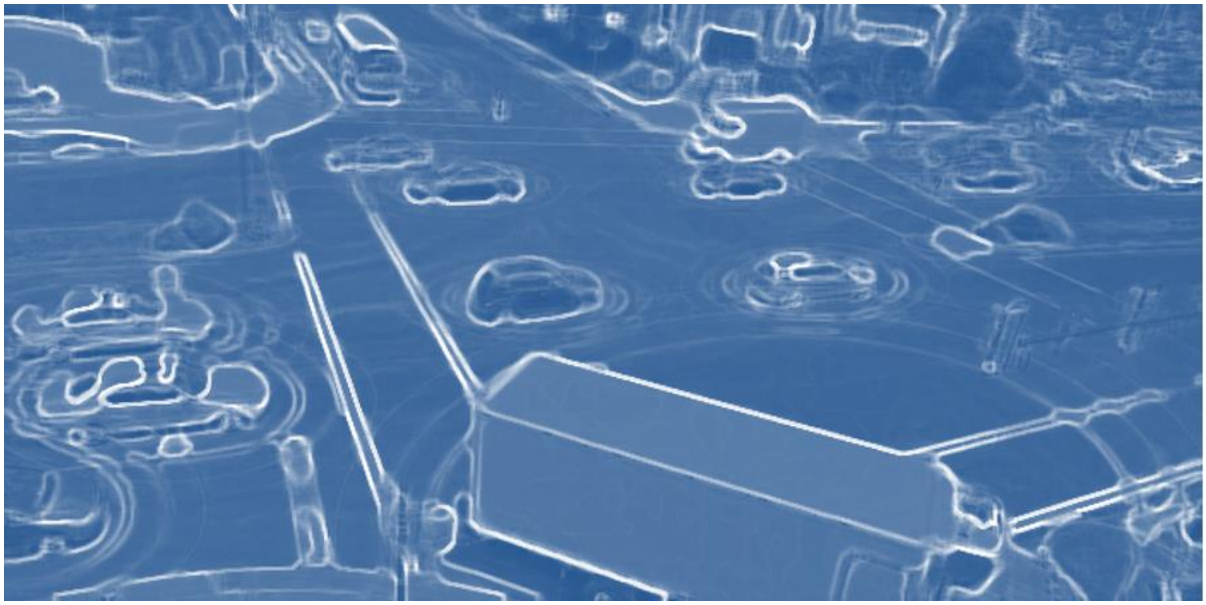


Test Control Interface Specification

Document Mnemonics:	TCIS
Revision:	[V0.4.0]
Revision Date:	4/10/2017

Table of Contents

1 Scope 4

2 References 4

2.1 Normative References 4
2.2 Informative References 5

3 Abbreviations 5

4 Test System 6

4.1 Architecture 6
4.2 Hardware equipment 6
4.2.1 Test System 7
4.3 DSRC radio 7
4.4 Interface Requirements 7
4.4.1 Test System Interface (TS ←→ SUT) 7
4.4.2 Interface to DSRC Radio (TS ←→ DSRC Radio) 7
4.4.3 Constraints 8

5 TCI Message Protocol 8

5.1.1 TS sends a request to SUT and receives a Response 8
5.1.2 SUT sends an unsolicited Indication to the TS 9
5.1.3 TS sends a request and receives information from the SUT 9
5.1.4 SUT sends an unsolicited Exception to the SUT 9
5.2 Transport Protocol 9

6 Test Control Interface Messages 10

6.1 Shared message structure 10
6.2 Test Control Interface Modules 11

7 Common TCI modules 11

7.1 TCIwsm module 11
7.1.1 Request messages 12
7.2 TCIip module 17
7.2.1 Request messages 17
7.3 Response, ResponseInfo, Indication and Exception messages 21
7.3.1 Response messages 21
7.3.2 Indication messages 21
7.3.3 ResponseInfo messages 22
7.3.4 Exception messages 23

8 TCI frames 25

8.1 TCI80211 frame 25
8.1.1 Supported use cases 25
8.1.2 Request Messages 25
8.1.3 Response messages 26
8.1.4 Indication messages 26
8.1.5 Exception messages 26
8.2 TCI16094 frame 26
8.2.1 Supported use cases 26
8.2.2 Request Messages 27

8.2.3 Response messages 28
8.2.4 Indication messages 28
8.2.5 ResponseInfo messages 29
8.2.6 Exception messages 29
8.3 TCI16093 frame 29
8.3.1 Supported use cases 29
8.3.2 Request messages 31
8.3.3 Response messages 32
8.3.4 Indication messages 32
8.3.5 ResponseInfo messages 33
8.3.6 Exception messages 33
8.4 TCI29451 frame 34
8.4.1 Request messages 35
8.4.2 Response messages 41
8.4.3 Indication messages 41
8.4.4 ResponseInfo messages 41
8.4.5 Exception messages 41
8.5 TCISutControl 41
8.5.1 Supported use cases 41
8.5.2 Request messages 42
8.5.3 Response messages 42
8.5.4 ResponseInfo messages 42
8.5.5 Exception messages 43

Appendix A: TCI protocol ASN.1 definition 44

Revision History 46
Open Issues 46

1 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.

2 References

2.1 Normative References

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

[1] WAVE802.11-TSS&TP (V0.5.0): "Conformance test specifications for Wireless Access in Vehicular Environments (WAVE) — 802.11 Test Suite Structure and Test Purposes (TSS & TP)". Revision date: 2/23/2016

[2] WAVEMCO-TSS&TP (V0.4.0): "Conformance test specifications for Wireless Access in Vehicular Environments (WAVE) — Multi-channel Operation Test Suite Structure and Test Purposes (TSS & TP)". Revision date: 2/22/2016

[3] WAVENS-TSS&TP (V0.6.0): "Conformance test specifications for Wireless Access in Vehicular Environments (WAVE) — Networking Services Test Suite Structure and Test Purposes (TSS & TP)". Revision date: 1/6/2016

[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: 2/22/2016

[5] J2945/1-TSS&TP (V0.3.0): "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: 2/26/2016

[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".

2.2 Informative References

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to 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".

3 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

4 Test System

4.1 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.

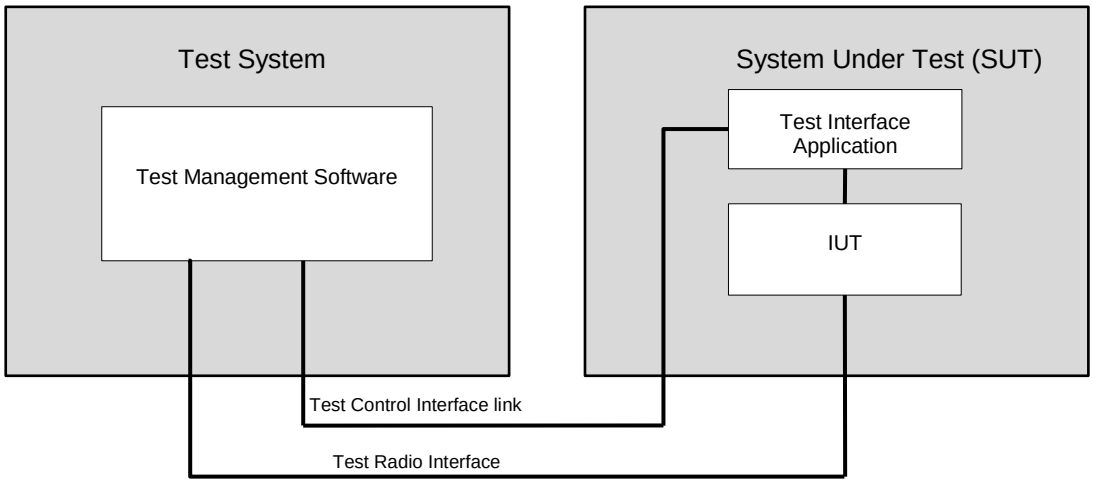


Figure 1: General Architecture

4.2 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 IP-based 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 as long as it 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 DSRC radio using a separate wired Ethernet link. The DSRC radio 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.

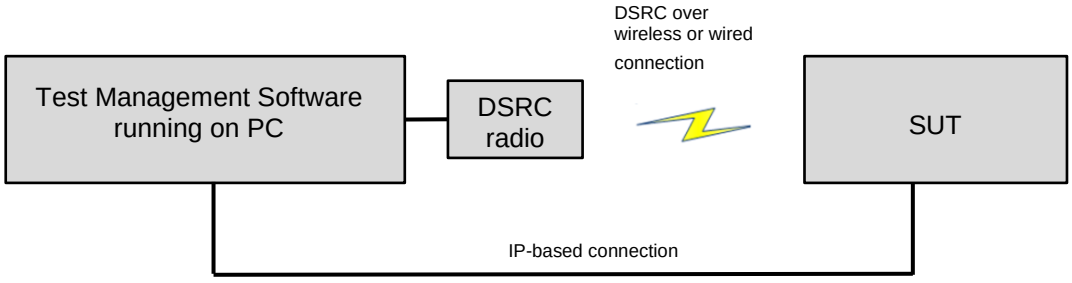


Figure 2: Test System Implementation

4.2.1 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 have to 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 in order 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 in regards to the DSRC wireless exchanges since the IPv4 protocol is not supported for DSRC over-the-air transmissions.

4.3 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].

4.4 Interface Requirements

4.4.1 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.

4.4.2 Interface to DSRC Radio (TS ←→ DSRC Radio)

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

- The SUT communicates to the DSRC radio using DSRC wireless protocol
- The DSRC radio translates the received WSM messages and sends them to the TS using UDP protocol.
- The DSRC radio receives UDP packets from the TS and transmits them as WSM over DSRC protocol.
- The conversion between the WSM and UDP protocol is performed as described in [6].

4.4.3 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.

5 TCI Message Protocol

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.

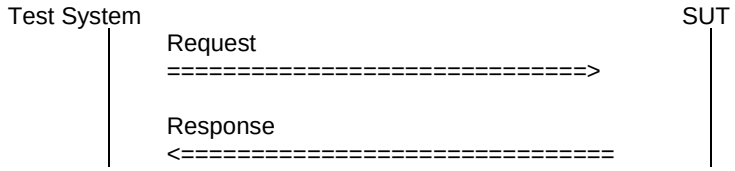
The message exchange format is laid out as follows

- **Request**: This message is initiated from the Test System to the SUT in order 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 in order 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:

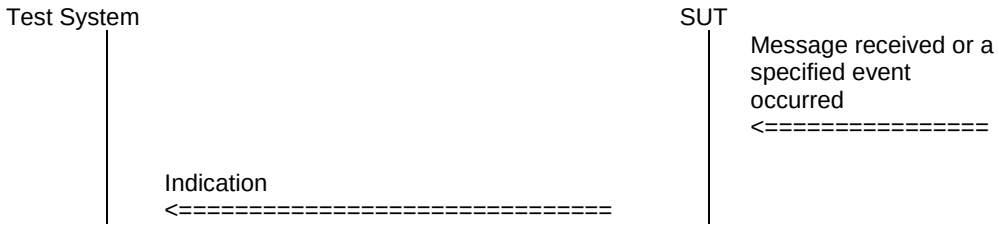
5.1.1 TS sends a request to SUT and receives a *Response*



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.

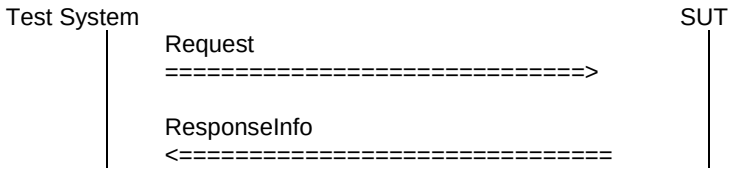
The response is actually an acknowledgement that the SUT received the test system's request and will be acting on it. It then executes the request. It is the TS that determines if the test passes or fails based on the result of the test.

5.1.2 SUT sends an unsolicited *Indication* to the TS



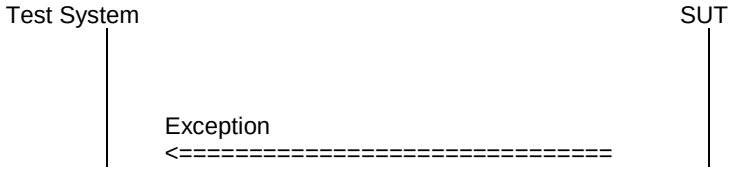
This communication exchange is initiated by the SUT. The SUT may send an unsolicited indication to the TS each time a packet is received and processed by the SUT or an event occurred on the SUT. Normally, the SUT will start (or stop) sending *Indications* after it is triggered by the TS. The TS never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

5.1.3 TS sends a request and receives information from the SUT



The TS needs to obtain information from the SUT, e.g. the IPv6 address of the DSRC 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.

5.1.4 SUT sends an unsolicited Exception to the SUT



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. All TCI messages are encoded using OER encoding. Note, that some TCI messages may contain a parameter containing a DSRC 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.

5.2 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

In order 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*.

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.

6 Test Control Interface Messages

6.1 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 1
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{
	TCI16093
	TCI16094
	TCI80211
	TCI29451
	TCISutControl
	}
		Current TCI frames defined in this specification.

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

```
TCI16093 ::= 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.

6.2 Test Control Interface Modules

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

Table 2 TCI protocol modules

Module (asn extensions omitted)	Description
TCIDispatcher	Root module aggregating all other frame specific messages
TCI16092	Frame and message definition used for testing 1609.2
TCI16093	Frame and message definition used for testing 1609.3
TCI16094	Frame and message definition used for testing 1609.4
TCI29451	Frame and message definition used for testing 2945/1
TCI80211	Frame and message definition used for testing 802.11
TCICommonTypes	Common types shared across TCI modules
TCIwsm	Request messages for sending and receiving WSM packets
TCIip	Request messages for sending and receiving IPv6 packets
TCISutControl	Device-level commands for controlling SUT
TCIEventHandling	Common event-handling types shared by other modules
TCIindication	Common indication messages shared by other modules

For example, several TCI frames trigger transmission of WSM. Those requests are defined in TCIwsm and included into corresponding TCI16093, TCI80211, etc by reference. Similarly, requests to transmit IPv6 packets are defined in TCIip and imported into TCI16093, TCI16094, etc. by reference.

7 Common TCI modules

This section describes common messages shared by TCI frames.

7.1 TCIwsm module

TCIwsm modules 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, channelIdentifier, dataRate, transmitPowerLevel, userPriority, 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 OER encoding. Due to encoding difference, the same parameters values may have different representation once encoded for transmission as WSM compared to TCI messages.

7.1.1 Request messages

7.1.1.1 SetInitialState

This request is used to set the SUT in initial condition. The initial condition defines the initial state in which the SUT has to 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].

7.1.1.2 SetWsmTxInfo

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

```
SetWsmTxInfo ::= SEQUENCE{
    psid                    Psid,
    radio                   RadioInterface,
    security                SecurityContext,
    channelIdentifier       ChannelNumber80211,
    timeslot                TimeSlot,
    dataRate                DataRate,
    transmitPowerLevel      TXpower80211,
    infoElementsIncluded    WaveElementsIncluded DEFAULT '000000000000000000000000'B,
    userPriority            UserPriority,
    destinationMACAddr      MACaddress    DEFAULT 'FFFFFFFFFFFF'H,
    expiryTime              INTEGER(0..18446744073709551615) OPTIONAL,
    channelLoad             Opaque 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.
channelIdentifier	Channel number as defined in 1609.3 [8].
timeslot	Time slot or continuous channel usage as defined in 1609.3 [8].
dataRate	Data rate as defined in 1609.3 [8].
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].
destinationMACAddr	Destination MAC address for the destination as defined in 1609.3 [8]. Default value set for broadcast transmissions.
expiryTime	Expiry time as defined in 1609.3 [8].This is an optional parameter.
channelLoad	Channel load as defined in 1609.3 [8].

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

7.1.1.3 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].

```
StartWsmTx ::= SEQUENCE{
    psid              Psid,
    radio             RadioInterface,
    repeatRate        RepeatRate, -- number of msg per 5 sec interval
    payload           Opaque,
```

```
    ...
}
```

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 interval. Additionally, it can be set to 0 for transmitting a single message.
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.

7.1.1.4 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,
    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.

7.1.1.5 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 DEFAULT 'FFFFFFFFFFFF'H,
    wsaChannelIdentifier     ChannelNumber80211,
    channelAccess            TimeSlot,
    repeatRate               RepeatRate, -- number of msg per 5 sec interval
    ipService                BOOLEAN,
    security
        SecurityContext (WITH COMPONENTS {
               contentType (mWSA)
        }),
    signatureLifetime        INTEGER(10..30000),
    infoElementIncluded      WaveElementsIncluded DEFAULT '00000000000000000000000'B,
    advertiserId             AdvertiserIdentifier OPTIONAL,
    serviceInfos             ServiceInfos,
    channelinfos             ChannelInfos,
    wra                      RoutingAdvertisement OPTIONAL,
    -- if the following parameters omitted, use the default values from the SUT MIB
```

```
    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 interval. 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.

7.1.1.6 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.

7.1.1.7 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 *WME-ChannelService* from 1609.3[8].

```
StartWsmRx ::= SEQUENCE{
    psid                    Psid,
    radio                   RadioInterface,
    channelIdentifier       ChannelNumber80211,
    timeSlot                TimeSlot,
    eventHandling           EventHandling,
    ...
}
```

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.

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.

7.1.1.8 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.

7.1.1.9 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.

7.1.1.10 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.

7.1.1.11 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,
    ...
```

```
}
```

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 includes 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].
channelLoad	Channel Load as defined in 1609.3 [8].

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

7.1.1.12 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.

7.2 TCIip module

TCIip modules 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.

7.2.1 Request messages

7.2.1.1 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 and contains:

```
Ipv6InterfaceInfo ::= SEQUENCE OF SEQUENCE {
    interfaceName     UTF8String(SIZE(1..255)), -- e.g. “eth0”,
    ipAddress         SEQUENCE OF IPv6Address, -- linked local, global, etc
    macAddress        MACaddress, -- MAC address for the network interface
    defaultGateway    IPv6Address, -- default gateway IPv6 address (assigned via WSA/WRA)
    primaryDns        IPv6Address, -- primary DNS IPv6 address (assigned via WSA/WRA)
    gatewayMacAddress MACaddress, -- gateway Mac address (assigned via WSA/WRA)
    ...
}
```

7.2.1.2 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,
    -- optional if the new IPv6 address value must be selected at random
    ...
}
```

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.*
ipAddresses	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.

7.2.1.3 StartIPv6Tx

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

```
StartIPv6Tx ::= SEQUENCE{
    radio             RadioInterface,
    interfaceName     UTF8String(SIZE(1..255)),
    destIpAddress     IpAddress,
    destPort          IpPort OPTIONAL,
    protocol          ENUMERATED { tcp, udp, icmp },
    repeatRate        RepeatRate OPTIONAL, -- number of msg per 5 sec interval
    eventHandling     EventHandling  (WITH COMPONENTS {..., eventFlag ('000000000'B) })
OPTIONAL,
    payload                 Opaque OPTIONAL,
    ...
}
```

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 message content.

7.2.1.4 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.

7.2.1.5 SendIpv6Ping

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.

```
SendIPv6Ping ::= StartIPv6Tx ( WITH COMPONENTS {
    radio,
    interfaceName,
    destIpAddress,
    destPort ABSENT,
    protocol (icmp),
    repeatRate OPTIONAL, -- number of msg per 5 sec interval
    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 interval. 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.

7.2.1.6 StartIPv6Rx

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

```
StartIPv6Rx ::= SEQUENCE{
    radio              RadioInterface,
    interfaceName      UTF8String(SIZE(1..255)),
    listenPort         IpPort,
    protocol           ENUMERATED { tcp (0), udp (1) },
    eventHandling      EventHandling
                       (WITH COMPONENTS {..., eventFlag ({eIpv6PktRx}) }) OPTIONAL,
    ...
}
```

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.

7.2.1.7 StopIPv6Rx

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

```
StopIPv6Rx ::= StartIPv6Rx ( 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.

7.3 Response, ResponseInfo, Indication and Exception messages

7.3.1 Response messages

The *Response* message is sent in response to the *Request*. It is defined in the *TCICommonType.asn* 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.

7.3.2 Indication messages

The *Indication* message is sent from the SUT to TS. It is defined in the *TCIindication.asn* module.

```
Indication ::= SEQUENCE{
    radio             RadioInterface,
    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 *TCIIndication.asn* module.

Table 22 Events which can trigger Indication messages

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

7.3.3 ResponseInfo messages

This message is used to retrieve configuration information from the SUT. It is defined in the *TCIresponseInfo.asn* 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.

7.3.4 Exception messages

Exception is a message sent from the SUT to the TS. It is used to report certain conditions to the TS. There is 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.

```
Exception ::= SEQUENCE{
    type       ExceptionType,
    id         ExceptionId OPTIONAL,
    module     Module OPTIONAL,
    text       ExceptionText 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

8 TCI frames

8.1 TCI80211 frame

8.1.1 Supported use cases

Use cases (UC) supported by TCI802.11 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	The SUT transmits a single or periodic WSMs	TS -> SUT SUT -> TS	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	request. StartWsmRx response
5	The SUT stops receiving WSMs	TS -> SUT SUT -> TS	request. StopWsmRx response

8.1.2 *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* corresponding to the *Request* message.

Most of these messages are imported from the common *TCIwsm* module.

Table 27 Listing of *Request* messages

Request Messages	MsgID	Explanation
SetInitialState	1	Request to configure SUT to the Initial state
StartWsmTx	2	Request to start transmission of WSMs
StopWsmTx	3	Request to stop transmission of WSMs
StartWsmRx	4	Request to start reception of WSMs
StopWsmRx	5	Request to stop reception of WSMs

8.1.2.1 SetInitialState

This request is used to set the SUT in initial condition. This request is defined in *TCIwsm*.

8.1.2.2 StartWsmTX

This request is used to initiate transmission of WSMs by the SUT. This request is defined in *TCIwsm*.

8.1.2.3 StopWsmTx

This request is used to cease transmission of WSMs by the SUT. This request is defined in *TCIwsm*.

8.1.2.4 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 *TCIwsm*.

8.1.2.5 StopWsmRX

This request is used to stop the SUT reception of messages and generation of *Indication* messages. This request is defined in *TCIwsm*.

8.1.3 *Response* messages

The *Response* message is sent in response to the *Request*. *Response* is defined in the *TCICommonTypes* module.

8.1.4 *Indication* messages

The *Indication* message is sent from the SUT to the TS indicating an occurrence of a predefined event. TCI80211 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 *TCIindication* module.

8.1.5 *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 *TCICommonTypes* module.

8.2 TCI16094 frame

8.2.1 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	Reserved		
8	The TS requests information from the SUT about the radio (0..3) used for IPv6 Communication	TS -> SUT SUT -> TS	request.GetIpv6InterfaceInfo response
9	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
10	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. SendIpv6Ping response
11	The SUT transmits single or periodic IPv6 packets	TS -> SUT SUT -> TS	request. StartIPv6Tx response
12	The SUT stops transmitting periodic IPv6 packets	TS -> SUT SUT -> TS	request. StopIPv6Tx response
13	The SUT receives IPv6 packets and sends event indications to the TS	TS -> SUT SUT -> TS	request. StartIPv6Rx response
14	The SUT stops receiving IPv6 packets	TS -> SUT SUT -> TS	request. StopIPv6Rx response

8.2.2 *Request* Messages

Table 29 lists all supported R*equest* messages supported in the *TCI16094* 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	3	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
SendIpv6Ping	9	Transmit a single ping message over IPv6 and receive ping echo from the remote host
StartIPv6Tx	10	Request to start transmission of IPv6 packets
StopIPv6Tx	11	Request to stop transmission of IPv6 packets
StartIPv6Rx	12	Request to start reception of IPv6 packets
StopIPv6Rx	13	Request to stop reception of IPv6 packets

8.2.2.1 SetInitialState
This request is used to set the SUT in initial condition. This request is defined in the *TCIwsm* module.

8.2.2.2 SetWsmTxInfo
This request is used to configure the SUT's WSM transmission parameters. This request is defined in the *TCIwsm* module.

8.2.2.3 StartWsmTX
This request is used to initiate transmission of WSMs by the SUT. This request is defined in the *TCIwsm* module.

8.2.2.4 StopWsmTx
This request is used to cease transmission of WSMs by the SUT. This request is defined in the *TCIwsm* module.

8.2.2.5 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 *TCIwsm* module.

8.2.2.6 StopWsmRX
This request is used to stop SUT reception of messages and generation of *indication* messages. This request is defined in the *TCIwsm* module.

8.2.2.7 GetIpv6InterfaceInfo
This request is used to requests IPv6 configuration from the SUT. This request is defined in the *TCIip* module.

8.2.2.8 SetIpv6Address
This request is used to change SUT IPv6 configuration. This request is defined in the *TCIip* module.

8.2.2.9 SendIpv6Ping
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 *TCIip* module.

8.2.2.10 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.

8.2.2.11 StopIPv6Tx
This request is used to cease transmission of IPv6 packets by the SUT. This request is defined in the *TCIip* module.

8.2.2.12 StartIPv6Rx
This request is used to initiate reception of IPv6 packets by the SUT. This request is defined in the *TCIip* module.

8.2.2.13 StopIPv6Rx
This request is used to cease reception of IPv6 packets by the SUT. This request is defined in the *TCIip* module.

8.2.3 *Response* messages

The *Response* message is sent in response to the *Request*. *Response* is defined in the *TCICommonTypes* module.

8.2.4 *Indication* messages

The *Indication* message is sent from the SUT to the TS indicating an occurrence of a predefined event. TCI16094 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 *TCIindication* module.

8.2.5 *ResponseInfo* messages

This message is used to retrieve configuration information from SUT.

```
Dot4ResponseInfo ::= ResponseInfo (WITH COMPONENTS {
    msgID,
    resultCode,
    info (WITH COMPONENTS {
        ipv6InterfaceInfo}  ) OPTIONAL, -- if exception reported, no InfoContent provided
    exception OPTIONAL
    })
```

8.2.6 *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 *TCICommonTypes* module.

8.3 TCI16093 frame

8.3.1 Supported use cases

Use cases (UC) supported by TCI16093 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
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
15	An exception occurred on SUT and reported to the TS	SUT -> TS	exception

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.

8.3.2 *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
delWsaProviderService	10	Request to delete a service provider from an existing WSA broadcast
addUserService	11	Request to add a user service
delUserService	12	Request to delete a user service
getIpv6InterfaceInfo	13	Request to SUT to report its IPv6 configuration
setIpv6Address	14	Request to SUT to set its IPv6 address
sendIpv6Ping	15	Request to SUT to send a ping (ICMP over IPv6)

8.3.2.1 SetInitialState

This request is used to set the SUT in initial condition. This request is defined in the *TCIwsm* module.

8.3.2.2 SetWsmTxInfo

This request is used to configure the SUT's WSM transmission parameters. This request is defined in the *TCIwsm* module.

8.3.2.3 StartWsmTX

This request is used to initiate transmission of WSMs by the SUT. This request is defined in the *TCIwsm* module.

8.3.2.4 StopWsmTx

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

8.3.2.5 StartWsaTxPerdiodic

This request is used to initiate transmission of WSA by the SUT. This request is defined in the *TCIwsm* module.

8.3.2.6 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 *TCIwsm* module.

8.3.2.7 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 *TCIwsm* module.

8.3.2.8 StopWsmRX
This request is used to stop the SUT's reception of messages and generation of *indication* messages. This request is defined in the *TCIwsm* module.

8.3.2.9 AddWsaProviderService
This request is used to add a provider service and update WSA. This request is defined in the *TCIwsm* module.

8.3.2.10 DelWsaProviderService
This request is used to removes a provider service and updates WSA. This request is defined in the *TCIwsm* module.

8.3.2.11 AddUserService
This request is used to add a user service to the SUT. This request is defined in the *TCIwsm* module.

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

8.3.2.13 GetIpv6InterfaceInfo
This request is used to retrieve IPv6 configuration from the SUT. This request is defined in the *TCIip* module.

8.3.2.14 SetIpv6Address
This request is used to set IPv6 address on the SUT. This request is defined in the *TCIip* module.

8.3.2.15 SendIpv6Ping
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 *TCIip* module.

8.3.3 *Response* messages

The *Response* message is sent in response to the *Request*. *Response* is defined in the *TCICommonTypes* module.

8.3.4 *Indication* messages

The *Indication* message is sent from the SUT to the TS indicating an occurrence of a predefined event. TCI16093 defines *Dot3Indication* as follows:

```
Dot3Indication ::= 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 the *TCIindication* 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.

8.3.5 *ResponseInfo* messages

This message is used to retrieve configuration information from the SUT. TCI16093 defines *Dot3ResponseInfo* as follows:

```
Dot3ResponseInfo ::= ResponseInfo (WITH COMPONENTS {
    msgID,
    resultCode,
    info (WITH COMPONENTS {
        ipv6InterfaceInfo}  ) OPTIONAL, -- if exception reported, no InfoContent provided
    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.

8.3.6 *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 *TCICommonTypes* module.

8.4 TCI29451 frame

Use cases supported by TCI29451 are listed in Table 40.

Table 40 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.ConfigureBsm response request.StartBsmTx response
3	The SUT stops transmitting periodic BSMs	TS -> SUT SUT -> TS	request.StopBsmTx response
4	The SUT starts receiving BSMs	TS -> SUT SUT -> TS	request.StartBsmRx response
5	The SUT stops receiving BSMs	TS -> SUT SUT -> TS	request.StopBsmRX response
6	Set a position for the SUT after turning off GPS input	TS -> SUT SUT -> TS TS -> SUT SUT -> TS	request.EnableGpsInput = false response request.SetPosition response
7	Change the position of the SUT after turning off GPS input	TS -> SUT SUT -> TS TS -> SUT SUT -> TS	request.EnableGpsInput = false response request.ChangePosition response
8	Change the speed of the SUT after turning off GPS input	TS -> SUT SUT -> TS TS -> SUT SUT -> TS	request.EnableGpsInput = false response request.ChangeSpeed response
9	Change the heading of the SUT after turning off GPS input	TS -> SUT SUT -> TS TS -> SUT SUT -> TS	request.EnableGpsInput = false response request.ChangeHeading response
10	Turn the brake pedal status of the SUT on or off	TS -> SUT SUT -> TS	request.EnableBrakePedalStatus response
11	Change the yaw rate of the SUT after turning off GPS input	TS -> SUT SUT -> TS TS -> SUT SUT -> TS	request.EnableGpsInput = false response request.ChangeYawRate response
12	Set the exterior lights status of the SUT	TS -> SUT SUT -> TS	request.SetExteriorLightsStatus response
13	Turn the GPS input of the SUT on or off	TS -> SUT SUT -> TS	request.EnableGpsInput response
14	Turn the brake availability of the SUT on or off	TS -> SUT SUT -> TS	request.EnableBrakeAvailability response
15	Turn congestion mitigation of the SUT on or off	TS -> SUT SUT -> TS	request.EnableCongestionMitigation response

16	Set the Temporary ID of the SUT	TS -> SUT SUT -> TS	request.SetTemporaryId response
17	Set the Message Count of the SUT	TS -> SUT SUT -> TS	request.SetMsgCount response
18	Set the vehicle event flags of the SUT	TS -> SUT SUT -> TS	Request.SetVehicleEventFlags response
19	Set the transmission of the SUT	TS -> SUT SUT -> TS	request.SetVehicleTransmission response
20	Set the availability of individual brake pedal status of the SUT	TS -> SUT SUT -> TS TS -> SUT SUT -> TS TS -> SUT SUT -> TS	request.EnableBrakeAvailability = true response request.EnableIndividualBrakePedalStatus = true response request.SetIndividualBrakePedalStatus response

8.4.1 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
setPosition	2	Set a position for the SUT, overwriting its current position
changePosition	3	Change the position of the SUT relative to its current position
changeSpeed	4	Change the speed of the SUT relative to its current speed
changeHeading	5	Change the heading of the SUT relative to its current heading
changeYawRate	6	Change the yaw rate of the SUT relative to its current yaw rate
enableGpsInput	7	Enable or disable GPS input to the SUT
setVehicleTransmission	8	Set the transmission state of the SUT, overwriting its current transmission
setExteriorLightsStatus	9	Set the exterior lights status of the SUT, overwriting its current light status
setVehicleEventFlags	10	Set the vehicle flags of the SUT, overwriting its current flags
enableIndividualBrakePedalStatus	11	Enable or disable the brake pedal status of the SUT
enableBrakeAvailability	12	Enable or disable the brake availability of the SUT
enableCongestionMitigation	13	Enable or disable the congestion mitigation on the SUT
setTemporaryId	14	Set the temporary ID of the SUT, overwriting the current ID
setMsgCount	15	Set the message count of the SUT, overwriting the current count
configureBsm	16	Configure the transmission parameter of BSMs from the SUT
startBsmTx	17	Begin transmission of BSMs
stopBsmTx	18	Stop transmission of BSMs
startBsmRx	19	Begin reception of BSMs
stopBsmRx	20	Stop reception of BSMs
setBrakePedal	21	Set the Brake Pedal status of the SUT, overwriting the current brake pedal status

8.4.1.1 SetInitialState

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

8.4.1.2 SetPosition

This request is used to set the position of the SUT. The definition of data units is adopted from [10].

```
SetPosition ::= SEQUENCE{
    latitude   Latitude,
    longitude  Longitude,
    elevation  Elevation
}
```

Parameters	Explanation
latitude	Parameter specifying the desired latitude of the SUT
longitude	Parameter specifying the desired longitude of the SUT
elevation	Parameter specifying the desired elevation of the SUT

The control of SUT position via TCI instead of GPS sensor must be enabled with TCI *EnableGpsInput* set to False. Then, the SUT position can be controlled via TCI messages *SetPosition*, *ChangePosition*, etc. The SUT position will remain in effect until it is changed either via another TCI *SetPosition*, *ChangePosition* or the SUT GNSS sensor is enabled via TCI *EnableGpsInput* set to True.

8.4.1.3 ChangePosition

This request is used to change the position of the SUT relative to its initial position at the time of the request. The definition of data units is adopted from [10].

```
ChangePosition ::= SEQUENCE{
    deltaLatitude     Latitude,
    deltaLongitude    Longitude,
    deltaElevation    Elevation
}
```

Parameters	Explanation
deltaLatitude	Parameter specifying the desired change in latitude of the SUT
deltaLongitude	Parameter specifying the desired change in longitude of the SUT
deltaElevation	Parameter specifying the desired change in elevation of the SUT

8.4.1.4 ChangeSpeed

This request is used to change the speed of the SUT. The definition of data units is adopted from [10].

```
ChangeSpeed ::= INTEGER(-8191..8191)
```

8.4.1.5 ChangeHeading

This request is used to change the heading of the SUT. The definition of data units is adopted from [10].

```
ChangeHeading ::= INTEGER(-28800..28800)
```

8.4.1.6 ChangeYawRate

This request is used to change the yaw rate of the SUT. The definition of data units is adopted from [10].

```
ChangeYawRate ::= INTEGER(-65534..65534)
```

8.4.1.7 EnableGpsInput

This request is used to enable or disable GPS input within the SUT.

```
EnableGpsInput ::= BOOLEAN
-- True – use GPS sensor to establish SUT position, speed, heading, etc
-- False – use data provided by TCI messages to set SUT position, speed, heading, etc
```

8.4.1.8 SetVehicleTransmission

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

```
SetVehicleTransmission ::= 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

8.4.1.9 SetExteriorLightsStatus

This request is used to set the exterior lights of the SUT.

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

Parameters	Explanation
lowBeamHeadlightsOn	Low beam headlights are turned on
highBeamHeadlightsOn	High beam headlights are turned on
leftTurnSignalOn	Left turn signal is turned on
rightTurnSignalOn	Right turn signal is turned on
hazardSignalOn	Hazard signal is turned on
automaticLightControlOn	Automatic light control is turned on
daytimeRunningLightsOn	Daytime running lights are turned on
fogLightOn	Fog light is turned on
parkingLightsOn	Parameter specifying the desired state of the external lights

8.4.1.10 SetVehicleEventFlags

This request configures the vehicle event flags of the SUT.

```
SetVehicleEventFlags ::= BIT STRING {
    eventHazardLights               (0),
    eventStopLineViolation          (1), -- Intersection Violation
    eventABSactivated               (2),
    eventTractionControlLoss        (3),
    eventStabilityControlActivated  (4),
    eventHazardousMaterials         (5),
    eventReserved1                  (6),
    eventHardBraking                (7),
    eventLightsChanged              (8),
    eventWipersChanged              (9),
    eventFlatTire                   (10),
    eventDisabledVehicle            (11), -- DisabledVehicle DF may also be sent
    eventAirBagDeployment           (12)
```

```
}
```

Parameters	Explanation
eventHazardLights	Parameter specifying whether a Hazard Light Event is occurring
eventStopLineViolation	Parameter specifying whether a Stop Line Violation Event is occurring
eventABSactivated	Parameter specifying whether an ABS Activated event is occurring
eventTractionControlLoss	Parameter specifying whether a Traction Control Loss event is occurring
eventStabilityControlActivated	Parameter specifying whether a Stability Control Activated event is occurring
eventHazardousMaterials	Parameter specifying whether a Hazardous Materials Event is occurring
eventReserved1	Parameter reserved for an event not explicitly included in the J2945.1 standard
eventHardBraking	Parameter specifying whether a Hard Braking event is occurring
eventLightsChanged	Parameter specifying whether a Lights Changes event is occurring
eventWipersChanged	Parameter specifying whether a Wipers Changed event is occurring
eventFlatTire	Parameter specifying whether a Flat Tire event is occurring
eventDisabledVehicle	Parameter specifying whether a Disabled Vehicle event is occurring
eventAirBagDeployment	Parameter specifying whether an Air Bag Deployment event is occurring

8.4.1.11 EnableIndividualBrakePedalStatus

Sets the brake pedal status of the SUT.

```
EnableBrakePedalStatus ::= BOOLEAN
```

8.4.1.12 EnableBrakeAvailability

This request sets the brake availability of the SUT.

```
EnableBrakeAvailability ::= BOOLEAN
```

8.4.1.13 EnableCongestionMitigation

This request sets the congestion mitigation of the SUT.

```
EnableCongestionMitigation ::= BOOLEAN
```

8.4.1.14 SetTemporaryId

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

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

8.4.1.15 SetMsgCount

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

```
SetMsgCount ::= INTEGER (0..127)
```

8.4.1.16 ConfigureBsm

This request configures the BSM transmission of the SUT. Refer to SetWsmTxInfo for more information on parameter settings.

```
ConfigureBsm ::= SetWsmTxInfo (WITH COMPONENTS {
        psid    (32),
        radio,
        security (WITH COMPONENTS { contentType (mBSM) }),
        channelIdentifier (172),
        timeslot (continuous),
```

```
        dataRate (r6Mbps-12BPSK),
        transmitPowerLevel (20),
        infoElementsIncluded ('00000000000000000000000'B),
        userPriority (7),
        destinationMACAddr ('FFFFFFFFFFFF'H),
        repeatRate ABSENT,
        payload ABSENT        -- Assumes BSM payload is generated by the SUT
    })
```

8.4.1.17 StartBsmTx

This request starts BSM transmission from the SUT. Refer to StartWsmTx for more information on parameter settings.

```
StartBsmTx ::= StartWsmTx (WITH COMPONENTS {
        psid (32),
        radio,
        repeatRate,           -- number of msg per 5 sec interval
        payload ABSENT        -- Assumes BSM payload is generated by the SUT
    })
```

8.4.1.18 StopBsmTx

This request stops BSM transmission from the SUT. Refer to StopWsmTx for more information on parameter settings.

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

8.4.1.19 StartBsmRx

This request starts BSM reception from the SUT. Refer to StartWsmRx for more information on parameter settings.

```
StartBsmRx ::= StartWsmRx (WITH COMPONENTS {
    psid  (32),
    -- PSID is optional if eventHandling.rxFlag is set to receive any WSM with PSID
    radio ( WITH COMPONENTS { ..., antenna ABSENT }),
    channelIdentifier,
    timeSlot,
    eventHandling
    })
```

8.4.1.20 StopBsmRx

This request stops BSM reception from the SUT. Refer to StopWsmRx for more information on parameter settings.

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

8.4.1.21 SetBrakePedal

This request sets the individual brakes on the SUT.

```
SetBrakePedal ::= BIT STRING {
    frontDriver             (0),
    forwardPassenger        (1),
    rearDriver              (2),
```

```
    rearPassenger            (3)
}
```

8.4.2 *Response* messages

The *Response* message is sent in response to the *Request*. *Response* is defined in the *TCICommonTypes* module.

8.4.3 *Indication* messages

The *Indication* message is sent from the SUT to the TS indicating an occurrence of a predefined event. TCI29451 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 *TCIindication* module.

8.4.4 *ResponseInfo* messages

TCI29451 does not use *ResponseInfo* messages.

8.4.5 *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 *TCICommonTypes* module.

8.5 TCISutControl

8.5.1 Supported use cases

Use cases (UC) supported by TCISutControl 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.SetTestId response

8.5.2 *Request* messages

Table 35 lists all supported R*equest* messages in the *TCISutControl*.

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

8.5.2.1 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 into a state where the CPU is halted and power draw is minimized.

8.5.2.2 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.

8.5.2.3 RequestSutAvailability

The TS sends to the SUT this message after restart or power up to determine the SUT status. 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 TS is not ready if it doesn't respond within the response timeout of **50ms** or includes the ResultCode = rcFailure.

8.5.2.4 RequestSutInfo

This request is used to obtain information version information from the SUT. This version information can be referenced in test reports and other test documentation.

8.5.2.5 SetTestId

The TS uses this request to send Test identifier to the SUT. The Test ID is a text string e.g. "TP-16093-WSM-MST-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 not 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.

8.5.3 *Response* messages

The *Response* message is sent in response to the *Request*. *Response* is defined in the *TCICommonTypes* module.

8.5.4 *ResponseInfo* messages

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

```
SutResponseInfo ::= ResponseInfo (WITH COMPONENTS {
    msgID,
    resultCode,
```

```
    info (WITH COMPONENTS {sutInfo}  ) OPTIONAL, -- if exception reported, no InfoContent
provided
    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 defined in the *TCI-responseInfo* ASN.1 module. Specific details for each type definition are listed in the ASN.1 specification referenced in Appendix A.

8.5.5 *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 *TCICommonTypes* 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 files are posted in github at the following location:

https://github.com/certificationoperatingcouncil/TCI_ASN1

TCIdispatcher.asn
- TCIMessage
- Frame

TCI16093.asn
- TCI16093
- Request
- MESSAGE-ID-AND-TYPE
- MessageTypes
- Dot3Indication
- Dot3ResponseInfo

TCI16094.asn
- TCI16094
- Request
- MESSAGE-ID-AND-TYPE
- MessageTypes
- Dot4Indication
- Dot4ResponseInfo

TCI29451.asn
- TCI29451
- Request
- MESSAGE-ID-AND-TYPE
- MessageTypes
- SetPosition
- ChangePosition
- ChangeSpeed
- ChangeHeading
- ChangeYawRate
- ConfigureBsm
- StartBsmTx
- StopBsmTx
- StartBsmRx
- StopBsmRx
- EnableGpsInput
- EnableBrakeAvailability
- EnableIndividualBrakePedalStatus
- EnableCongestionMitigation
- SetTemporaryId
- SetMsgCount
- SetVehicleEventFlags
- SetVehicleTransmission
- SetBrakePedal
- SetExteriorLightsStatus
- D2945Indication

TCI80211.asn
- TCI80211
- Request
- MESSAGE-ID-AND-TYPE
- MessageTypes
- Dot11Indication

TCIEventHandling.asn
- EventHandling
- RxFlag
- EventFlag
- SecurityFlag

TCIindication.asn
- Indication
- Event
- EventParams
- Pdu
- ServiceParameters
- WsmParameters
- IpParameters
- D80211Parameters
- SecResultParams
- SecurityResultCode

TCIip.asn
- GetIPv6InterfaceInfo
- SetIPv6Address
- StartIPv6Tx
- StopIPv6Tx
- SendIPv6Ping
- StartIPv6Rx
- StopIPv6Rx

TCIresponseInfo.asn
- ResponseInfo
- InfoContent
- Dot11PhyType
- Dot4StationConfigEntry
- Dot3StationConfigEntry
- Ipv6InterfaceInfo
- SutInfo
- VersionInfoBlock

TCISutControl.asn
- TCISutControl
- Request
- MESSAGE-ID-AND-TYPE
- Request
- MessageTypes
- Shutdown

Restart
RequestSutAvailability
RequestSutInfo
SetSutId
SutResponseInfo

TCIwsm.asn
SetInitialState
SetWsmTxInfo
StartWsmTx
StopWsmTx
AddUserService
DelUserService
StartWsmRx
StopWsmRx
StartWsaTxPerdiodic
StopWsaTxPeriodic
AddWsaProviderService
DelWsaProviderService
AddUserService
DelUserService

ContentType
SignerIdentifierType
SecurityContext
WaveElementsIncluded
UserRequestType
WsaType
ServiceInfos
ServiceInfo
ChannelOptions
RepeatRate
IPAddress
PduData

TCICommonTypes.asn
Antenna
DataRate
Exception
ExceptionId
ExceptionText
ExceptionType
HashedId8
IpAddress
IpPort
Module
MsgID
Opaque
PduData
PduType
Psid
Radio
RadioInterface
RCPI
RepeatRate
Response
ResultCode
Time64
TimeSlot
UdpPort
UserPriority
VarLengthNumber

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

Revision History

V0.1.0	March 21, 2016	Initial Draft
V0.2.0	June 1, 2016	* Some editorial revisions to address industry feedback. * Link to github is added.
V0.3.0	July 19, 2016	* Editorial changes based on comments from Noblis
V0.4.0	April 10, 2017	* Updates to the TCI29451 and TCISutControl frames * Removed TCI16092 frame which is redundant with TCI16093 * Add explanation about test setup * Editorial changes to match updated ASN.1

Open Issues

None

End of Document