



Intelligent Transport Systems (ITS); Architecture of conformance validation framework

Reference

RTR/ITS-00340

Keywords

architecture, conformance, ITS, testing

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Important notice

The present document can be downloaded from:

<http://www.etsi.org>

The present document may be made available in electronic versions and/or in print. The content of any electronic and/or print versions of the present document shall not be modified without the prior written authorization of ETSI. In case of any existing or perceived difference in contents between such versions and/or in print, the only prevailing document is the print of the Portable Document Format (PDF) version kept on a specific network drive within ETSI Secretariat.

Users of the present document should be aware that the document may be subject to revision or change of status.

Information on the current status of this and other ETSI documents is available at

<http://portal.etsi.org/tb/status/status.asp>

If you find errors in the present document, please send your comment to one of the following services:

http://portal.etsi.org/chaicor/ETSI_support.asp

Copyright Notification

No part may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm except as authorized by written permission of ETSI.

The content of the PDF version shall not be modified without the written authorization of ETSI.

The copyright and the foregoing restriction extend to reproduction in all media.

© European Telecommunications Standards Institute 2014.

All rights reserved.

DECT™, **PLUGTESTS™**, **UMTS™** and the ETSI logo are Trade Marks of ETSI registered for the benefit of its Members.
3GPP™ and **LTE™** are Trade Marks of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners.
GSM® and the GSM logo are Trade Marks registered and owned by the GSM Association.

Contents

Intellectual Property Rights	5
Foreword.....	5
Introduction	5
1 Scope	6
2 References	6
2.1 Normative references	6
2.2 Informative references.....	6
3 Abbreviations	7
4 Test platform overview	8
4.1 Constraints and requirements	8
4.2 General architecture	8
5 Hardware equipment	9
5.1 PC.....	9
5.2 G5 adapter box	10
6 Codecs	10
6.1 Introduction	10
6.2 Advanced details of implementation	11
7 Test Adapter	12
7.1 Introduction	12
7.2 Lower Tester	13
7.2.1 Overview	13
7.2.2 Advanced details of implementation	15
7.2.3 Extensibility of the test adapter.....	17
7.2.4 Adapter Control primitives	17
7.3 Platform Adapter	17
7.4 Upper Tester.....	17
Annex A: Codecs Source Code.....	19
Annex B: Test Adapter Source Code	20
Annex C: Upper Tester Message Format.....	21
C.1 Common Upper Tester Primitives.....	21
C.1.1 UtInitialize.....	21
C.1.2 ChangePosition.....	22
C.2 CAM Upper Tester Primitives	23
C.2.1 ChangeCurvature.....	23
C.2.2 ChangeSpeed	23
C.2.3 SetAccelerationControlStatus.....	24
C.2.4 SetExteriorLightsStatus	25
C.2.5 ChangeHeading	26
C.2.6 SetDriveDirection.....	26
C.2.7 ChangeYawRate	27
C.2.8 CamEventIndication.....	27
C.2.9 SetStationType	27
C.2.10 SetVehicleRole	28
C.2.11 SetEmbarkationStatus	29
C.2.12 SetPtActivation.....	29
C.2.13 SetDangerousGoods	30
C.2.14 SetDangerousGoodsExtended	31
C.2.15 SetLightBarSiren	31

C.3	DENM Upper Tester Primitives	32
C.3.1	GenerateDenmEvent	32
C.3.2	UpdateDenmEvent	34
C.3.3	TerminateDenmEvent	35
C.3.4	DenmEventIndication.....	35
C.4	GeoNetworking Upper Tester Primitives.....	36
C.4.1	GenerateGeoUnicast.....	36
C.4.2	GenerateGeoBroadcast.....	36
C.4.3	GenerateGeoAnycast.....	37
C.4.4	GenerateSHB.....	38
C.4.5	GenerateTSB	38
C.4.6	GnEventIndication.....	39
C.5	IPv6OverGeoNetworking Upper Tester Primitives	39
C.5.1	SendIPv6Message	39
C.5.2	GetInterfaceInfos.....	40
C.5.3	Gn6EventIndication.....	40
C.6	BTP Upper Tester Primitives	41
C.6.1	GenerateBtpA.....	41
C.6.2	GenerateBtpB	41
C.6.3	BtpEventIndication.....	42
Annex D:	Example of Test Platform implementation.....	43
Annex E:	Complete Test Adapter class diagram	47
Annex F:	Bibliography	48
History		50

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (<http://ipr.etsi.org>).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Foreword

This Technical Report (TR) has been produced by ETSI Technical Committee Intelligent Transport Systems (ITS).

Introduction

In response to EC mandate M/453 [i.14], ETSI Technical Committee (TC) ITS has standardized base and test specifications for ITS protocols. In a next step a prototype TTCN-3 test system was built and validated. The present document and its related TR 103 061 series [i.3] to [i.7], describe the design and validation of the prototype TTCN-3 test system.

The action described in the present document has supported the implementation of ITS standards by:

- Making available validated and standardized test specifications and thus enabling the application of reliable certification schemes.
- Executing conformance validation framework against real Implementations Under Test (IUTs) from industry and thus providing these companies a conformance assessment of their implementations. During the lifetime of this action, the conformance validation framework was as well provided at ITS Cooperative Mobility Services Interoperability events.
- Releasing all software as open source and thus allowing industry to build and run their own conformance validation framework.

1 Scope

The present document provides a description of the architecture of the ITS conformance validation framework, including definition of the test environment, codec and test adapter. It provides, as well, all the necessary source code to build and run the ITS conformance validation framework.

The ITS conformance validation framework integrates the test suites TS 102 871-3 [i.9], TS 102 868-3 [i.10], TS 102 869-3 [i.11], TS 102 870-3 [i.12] and TS 102 859-3 [i.13]. The results of the validation are described in the TR 103 061 series [i.3] to [i.7].

2 References

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the reference document (including any amendments) applies.

Referenced documents which are not found to be publicly available in the expected location might be found at <http://docbox.etsi.org/Reference>.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

2.1 Normative references

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

Not applicable.

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 ES 201 873-5 (V4.5.1): "Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 5: TTCN-3 Runtime Interface (TRI)".
- [i.2] ETSI EG 201 015 (V2.1.1): "Methods for Testing and Specification (MTS); Standards engineering process; A handbook of validation methods".
- [i.3] ETSI TR 103 061-1 (V1.2.1): "Intelligent Transport Systems (ITS); Testing; Part 1: Conformance test specifications for Co-operative Awareness Messages (CAM); CAM validation report".
- [i.4] ETSI TR 103 061-2 (V1.2.1): "Intelligent Transport Systems (ITS); Testing; Part 2: Conformance test specifications for Decentralized Environmental Notification basic Service Messages (DENM); DENM validation report".
- [i.5] ETSI TR 103 061-3 (V1.2.1): "Intelligent Transport Systems (ITS); Testing; Part 3: Conformance test specifications for Geographical addressing and forwarding for point-to-point and point-to-multipoint communications; GeoNetworking validation report".
- [i.6] ETSI TR 103 061-4 (V1.1.1): "Intelligent Transport Systems (ITS); Testing; Part 4: Conformance test specification for GeoNetworking Basic Transport Protocol (BTP); GeoNetworking BTP validation report".
- [i.7] ETSI TR 103 061-5 (V1.1.1): "Intelligent Transport Systems (ITS); Testing; Part 5: IPv6 over GeoNetworking validation report".

- [i.8] IEEE 802.11p: "IEEE Standard for Local and Metropolitan Area Networks - Specific requirements; Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications; Amendment 6: Wireless Access in Vehicular Environments".
- [i.9] ETSI TS 102 871-3(V.1.2.1): " Intelligent Transport Systems (ITS); Testing; Conformance test specifications for GeoNetworking ITS-G5; Part 3: Abstract Test Suite (ATS) and Protocol Implementation eXtra Information for Testing (PIXIT)".
- [i.10] ETSI TS 102 868-3 (V.1.2.1): "Intelligent Transport Systems (ITS); Testing; Conformance test specifications for Co-operative Awareness Messages (CAM); Part 3: Abstract Test Suite (ATS) and Protocol Implementation eXtra Information for Testing (PIXIT)".
- [i.11] ETSI TS 102 869-3 (V.1.3.1): "Intelligent Transport Systems (ITS); Testing; Conformance test specifications for Decentralized Environmental Notification Messages (DENM); Part 3: Abstract Test Suite (ATS) and Protocol Implementation eXtra Information for Testing (PIXIT)".
- [i.12] ETSI TS 102 870-3 (V.1.1.1): "Intelligent Transport Systems (ITS); Testing; Conformance test specifications for Geonetworking Basic Transport Protocol (BTP); Part 3: Abstract Test Suite (ATS) and Protocol Implementation eXtra Information for Testing (PIXIT)".
- [i.13] ETSI TS 102 859-3 (V.1.2.1): "Intelligent Transport Systems (ITS); Testing; Conformance test specifications for Transmission of IP packets over Geonetworking; Part 3: Abstract Test Suite (ATS) and Protocol Implementation eXtra Information for Testing (PIXIT)".
- [i.14] EC mandate M/453: "Standardisation mandate addressed to CEN, CENELEC and ETSI in the field of Information and Communication Technologies to support the interoperability of co-operative systems for Intelligent Transport in the European Community".

3 Abbreviations

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

AC	Adapter Control
ACC	Adaptive Cruise Control
API	Application Programming Interface
ASN	Abstract Syntax Notation
ATS	Abstract Test Suite
BTP	Basic Transport Protocol
CAM	Cooperative Awareness Message
CC	Cruise Control
DENM	Decentralized Environmental Notification Message
ETH	Ethernet
GN	GeoNetworking
HB	High Beam
IP	Internet Protocol
ITS	Intelligent Transportation Systems
IUT	Implementation Under Test
JDK	Java™ Development Kit
KAF	Keep Alive Functionality
LB	Low Beam
LT	Left Turn
MAC	Media Access Control
OS	Operating System
OSI	Open Systems Interconnection model
PC	Personal Computer
Pcap	Packet Capture
PDU	Protocol Data Unit
PICS	Protocol Implementation Conformance Statement
RT	Right Turn
SHB	Single Hop Broadcast
SUT	System Under Test

TA	Test Adapter
TC	Test Cases
TCI	TTCN-3 Control Interface
TP	Test Purposes
TRI	TTCN-3 Runtime Interface
TSB	Topology Scoped Broadcast
TTCN-3	Testing and Test Control Notation 3
UDP	User Datagram Protocol
UT	Upper Tester

4 Test platform overview

4.1 Constraints and requirements

The purpose of the ITS test platform is to provide a reliable set of software and hardware equipments that can be used to validate TTCN-3 abstract test suites (ATS) developed in ETSI.

The architecture of this test platform has been designed with respect to the following constraints:

- to be compatible with the requirements expressed in the validation handbook (EG 201 015 [i.2]);
- to be independent of the platform used to implement the test system;
- to be independent of the TTCN-3 tool provider;
- to be configurable and customizable;
- to provide tools and well defined interfaces to system under test (SUT), allowing test automation;
- to be easily extensible for future ITS protocols;
- to provide generic components that can be reused in other test platforms.

In order to ensure independence of hardware platforms, all software pieces running on the test platform have been implemented using Java™ language, using generic and widely used libraries.

Test tool independence has been achieved by isolating the tool specific interfaces from core functionalities of the platform. Adapting the current platform to a different test tool would only require the implementation of a very simple piece of software mapping tool-specific functions to generic functions defined in this project.

In addition, great care has been taken to separate ITS specific functionalities from generic test platform tasks in order to provide a maximum number of reusable components for future test platforms.

4.2 General architecture

Typically a TTCN-3 test platform is composed of four different components:

- The TTCN-3 test tool providing necessary software to execute the abstract test suites.
- The hardware equipment supporting TTCN-3 test execution and adaptation to SUTs.
- The codecs which convert protocol messages into their abstract TTCN-3 representation.
- The Test Adapter (TA) implementing interfaces with the device under test.

The interaction of these components is described in figure 1.

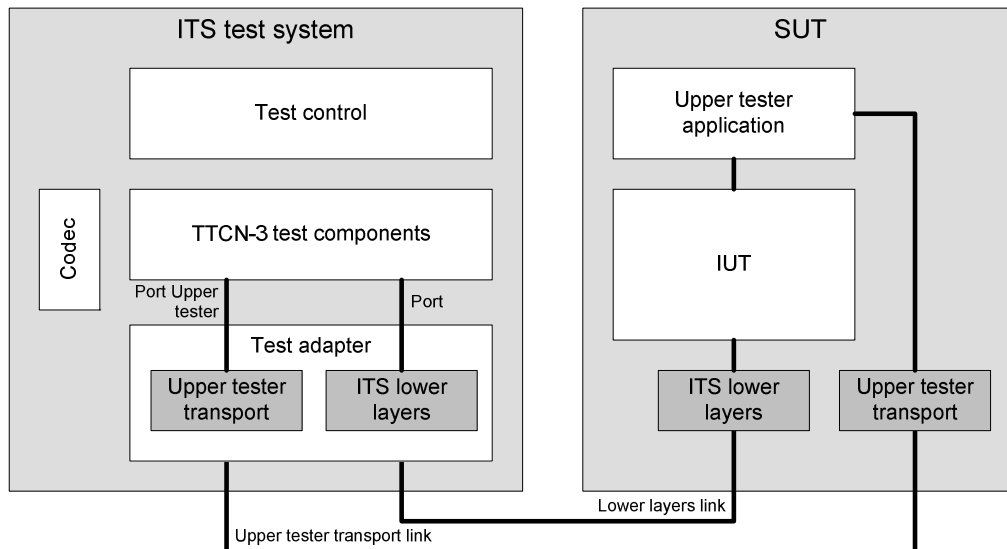


Figure 1: General architecture

The TTCN-3 test tools are usually provided by commercial companies and their description is out of the scope of the present document. The implementation details of the other components are described in the following clauses.

5 Hardware equipment

5.1 PC

The main hardware component of the ITS test platform is a standard PC. Its role is to host the execution of the test suites using a commercial TTCN-3 test tool.

Whatever operating system is installed on the computer, it is necessary to ensure that the following points are taken into account:

- No firewall interference with traffic generated by the Test System and/or SUT.
- Excellent time synchronization between the SUT and the test system.
- Test system processes (especially the test adapter) have to be granted unrestricted control to telecommunication hardware.

Time synchronization is maybe the most critical point to be checked before starting any test session, as it can be the source of strange SUT behaviour and generate incoherent results. Indeed, most ITS 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 ahead in time, all messages it sends will be considered either as coming from the future or from a very old date, and be discarded.

This PC is equipped with two network cards, one being used for ITS communication with SUT (lower layers link), the other one being used for exchanging upper tester messages (upper tester transport link). Separating these two communications on different hardware interfaces is not an absolute necessity, but it is a good practice and it ensures that there will be no interaction between the flows.

The communication between the SUT and the test system is achieved through Ethernet if the SUT supports it or using a G5 adaptation box.

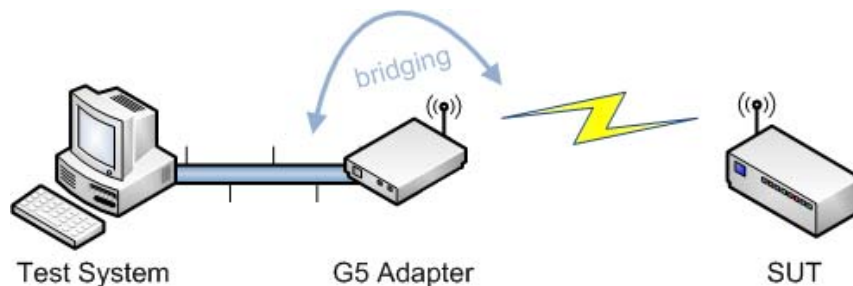


Figure 2: Communication via G5 adaptation box

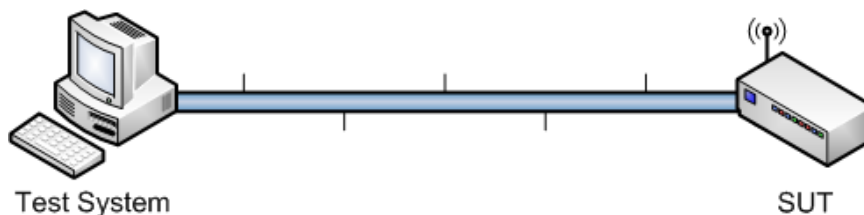


Figure 3: Communication via Ethernet

5.2 G5 adapter box

The ITS protocol stack makes use of G5 radio protocol in order to establish communication between ITS devices. To achieve G5 connectivity, a dedicated hardware equipment needs to be added to the test platform. The role of this adaptation box is to handle all radio-related tasks transparently and to act as a bridge for the test system.

Cohda WirelessTM MK2 has been chosen to fulfil this task. This device is fully IEEE 802.11p [i.8] compliant and provides as well an Ethernet interface so that it can be used as a transparent bridge between the test system and the SUT, as depicted in figure 2.

To transfer frames received on the Ethernet interface to the radio interface and vice versa, it is necessary to install and execute a small bridge application on the MK2. Only the frame featuring a specific ethertype (0x0707 by default) will be transferred from one interface to the other, so that only desired traffic will cross the bridge.

6 Codecs

6.1 Introduction

The codec entity is responsible for the encoding and decoding of TTCN-3 abstract values into bitstrings suitable to be sent to the System Under Test (SUT).

In order to simplify implementation and to ease the maintenance, coding and decoding tasks are handled by several codecs:

- One independent codec package per tested protocol;
- One codec package for TTCN-3 types that do not correspond to real protocol messages. It includes for example all auxiliary types used to carry information to/from Test Adapter, like the ones defined in TestSystem modules (CoapInd, CoapReq, etc.);
- One generic codec package available for handling default codec operation non related to any specific protocol. These codecs will be used if no protocol-specific codec exists for one type.

6.2 Advanced details of implementation

Figure 4 gives an overview of the relations between the different Java™ classes implementing the codecs. Connection with the tool-dependent classes is realized through the `ITERquired` interface and the associated factory class.

Each codec is responsible for correctly encoding and decoding one specific type and implements the `ICodec` interface.

Selection of correct codec for encoding or decoding a message at runtime is managed by the `CodecFactory` class, via the `getCodec()` method. This method will select the appropriate codec based on three parameters:

- the `type name`;
- the `encoding` as specified in TTCN-3 modules using "with encode" statements;
- the `type class` (record, union, etc.).

The rules for selecting the correct codec are the following:

- 1) If a codec is registered for `type name` in the package corresponding to `encoding`, then select this codec and call `encode()` or `decode()` method;
- 2) Otherwise, if a codec is registered for `type class` in the package corresponding to `encoding`, then select this codec and call `encode()` or `decode()` method;
- 3) Otherwise, use codec corresponding to `type class` in `generic` package.

This design provides both flexibility and easy extensibility. For most protocols, the "generic" codec package will handle most of the encoding and decoding operations. Specific encoding processes can be handled case by case by adding minimal codecs and registering them in the `CodecFactory`.

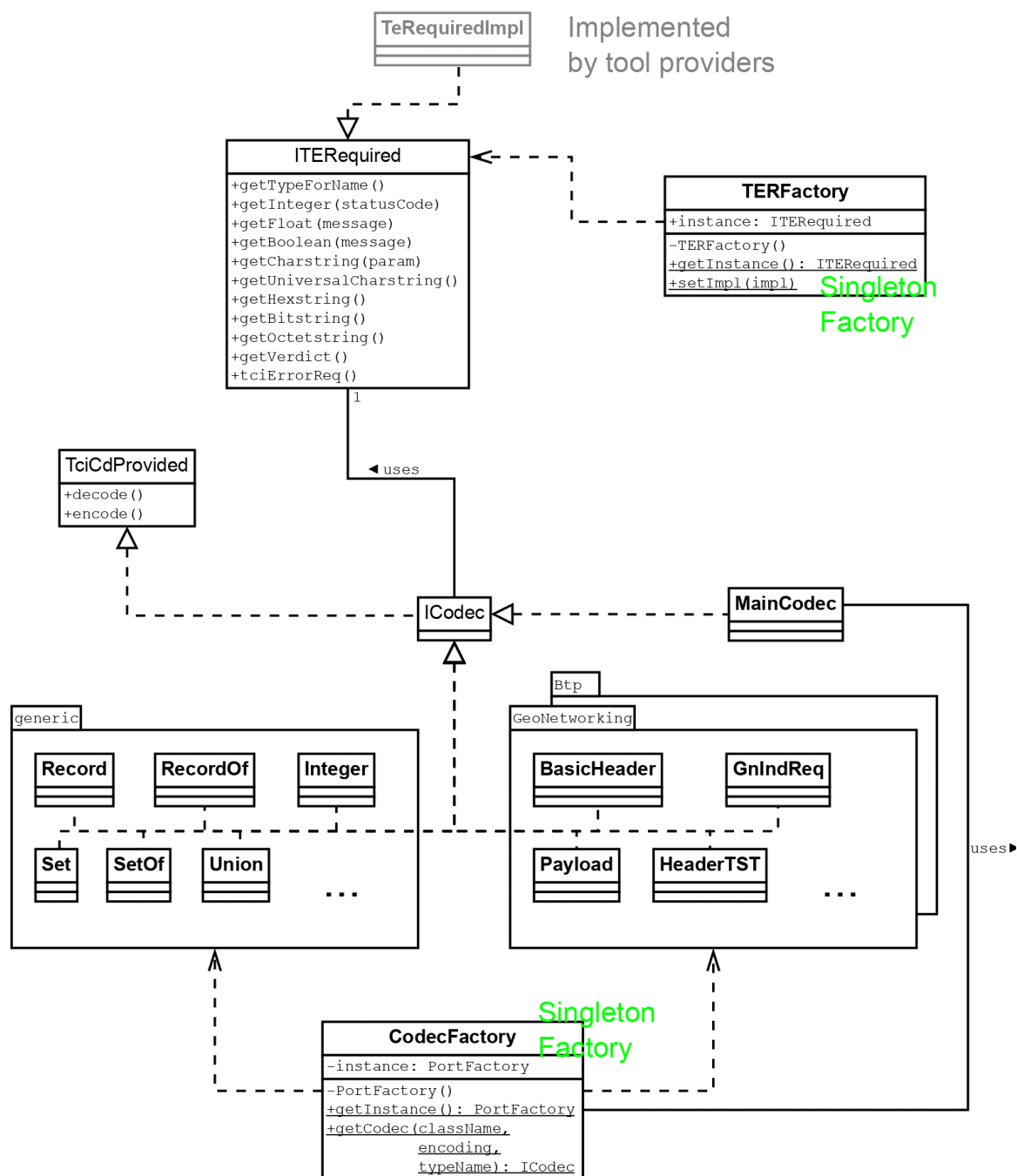


Figure 4: Codec class diagram

7 Test Adapter

7.1 Introduction

The test adapter conceptually splits into three parts:

- a lower test adapter;
- a TTCN-3 platform adapter implementing timers;
- an upper test adapter.

7.2 Lower Tester

7.2.1 Overview

TTCN-3 test suites are usually focusing on a single protocol layer and designed to be executed against real implementations (IUT). However, it is unusual to find standalone implementations as they are commonly integrated as an internal component of a physical device (SUT).

The purpose of a lower test adapter is to prepare and adapt the protocol messages used by TTCN-3 test suites so that they can be transmitted successfully to the SUT. One way to achieve this goal is for example to implement lower layers and encapsulate protocol messages accordingly. For instance, CAM and DENM messages need to be encapsulated in BTP datagrams, themselves encapsulated into GeoNetworking packets, and transmitted over G5 radio link. The higher up the IUT is located in the OSI stack, the more complex is the test adapter.

TTCN-3 test suites send and receive protocol messages via TTCN-3 communication ports. For each of these ports defined in the test suites, a corresponding port entity needs to be implemented in the test adapter using standardized TRI interface (ES 201 873-5 [i.1]). To provide maximum flexibility and allow for extensibility, the test adapter ports of the ITS test platform have been designed with the following constraints:

- For each port family, the lower stack can be configured using test adapter parameters (see annex D). As a consequence it is possible to dynamically define what will be the lower layers used to communicate with SUT, and how protocol messages will be encapsulated.
- All the instances of ports are independent.
- Behaviour of ports and lower layers can be dynamically modified by using predefined AC (Adapter Control) primitives directly sent from TTCN-3 script using dedicated port AcPort. For example, the AC primitive 'startBeaconing' requests the test adapter to start sending beacons.

The test adapter implementation mainly features `Port` and `Layer` objects. The relationship and interactions between these objects will be further detailed in clause 7.2.2, but it is important to notice the main differences between these objects, as misunderstanding their roles can lead to confusion:

- `Port` objects are the counterpart of TTCN-3 communication ports.
- `Layer` objects implement the minimal functionalities of a protocol layer and provide facilities for encapsulating or decapsulating packets.
- `Port` objects are configured with a lower stack composed of cascading `Layer` objects.
- For a same protocol layer, `Layer` objects usually implement more functionalities than `Port` objects.

Figures 5 and 6 show an example of interactions between these objects respectively when sending and receiving a message. The port described in this example is a CAM port configured with a BTP/GN/G5 lower stack.

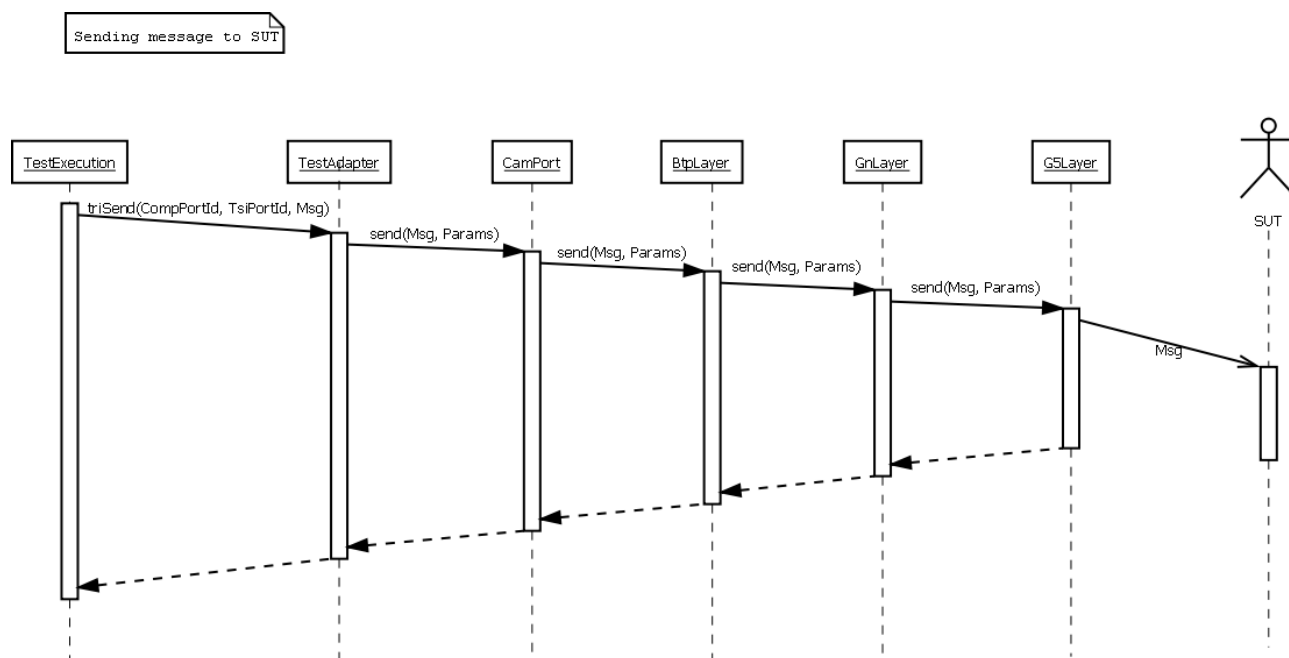


Figure 5: Message sending sequence diagram

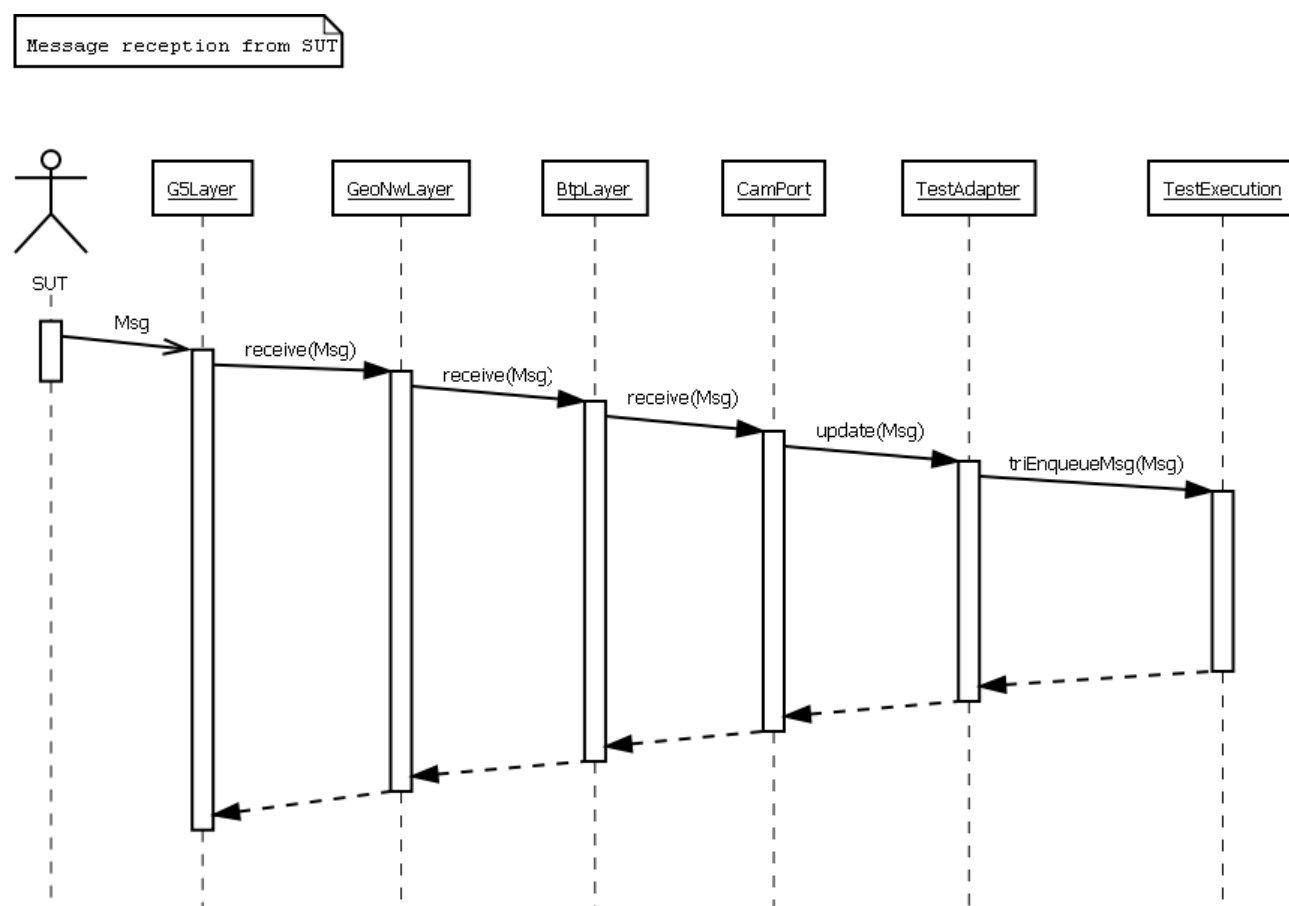


Figure 6: Message reception sequence diagram

7.2.2 Advanced details of implementation

Figure 7 presents the simplified class diagram of the test adapter. For better readability, auxiliary classes such as factories are not represented. The complete class diagram, also featuring design pattern indications, can be found in annex E.

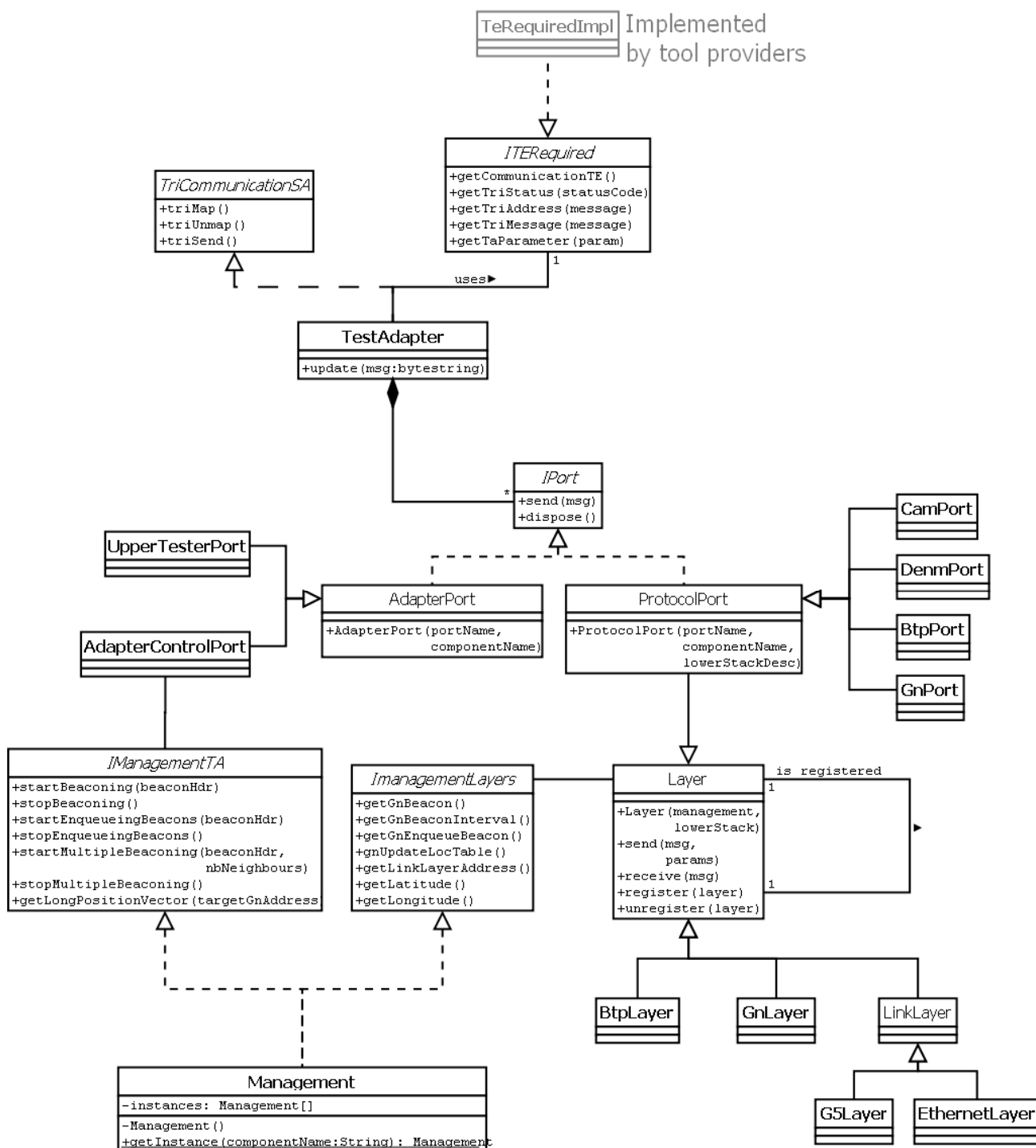


Figure 7: Test Adapter class diagram

The main class of this implementation is the **TestAdapter** class. It implements the standardized interface **TriCommunicationSA** and uses the **ITeRequired** interface implemented by test tool providers. These interfaces are part of the TRI API defined in ES 201 873-5 [i.1]. All other classes of this implementation are completely TRI-agnostic. The main purpose of the **TestAdapter** class is to instantiate and manage the different ports.

There are two different kinds of ports:

- Protocol ports realizing communication between TTCN-3 and SUT.
- Adapter ports which are used for test adapter configuration and upper tester communication.

Upon protocol port initialization, lower layers are instantiated in cascade and chained as depicted in figure 8, based on lower stack description. Figure 8 also illustrates the usage of "Factory" design patterns for instantiating `Layer` and `Port` objects. Each `Layer` is responsible for encapsulating and decapsulating packets and transmitting result to lower and upper layer using `send()`/`receive()` methods.

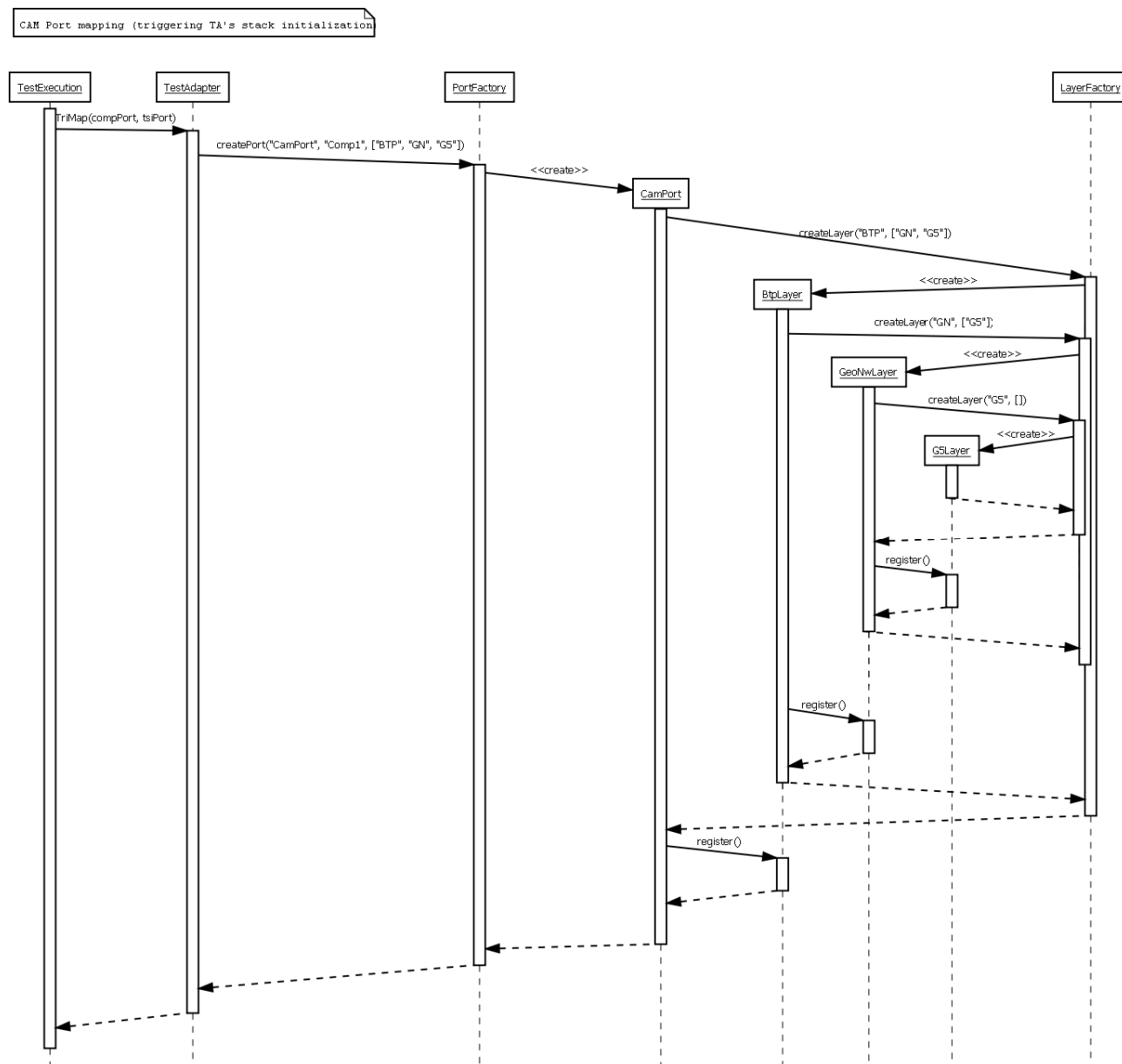


Figure 8: Port initialization sequence diagram

Currently the following layers have been implemented:

- GnLayer: basic functionalities of GeoNetworking layer, including beaconing.
- BtpLayer.
- EthernetLayer. It is important to note that this class requires the usage of the external library JnetPcap for capturing and injecting Ethernet frames.

- G5Layer.

The Management class, implementing IManagementTA and IManagementLayers interfaces is used for handling the dynamic configuration of Layer objects. It is directly linked to the AdapterControlPort and implements the AC primitives defined in the TTCN-3 test suites. It is important to notice that this class is implemented using a "Multiton or Multiple Singleton" design pattern: one single instance of this class can be instantiated per TTCN-3 component.

7.2.3 Extensibility of the test adapter

The test adapter can be extended in several ways. The first option is to add new protocol layers by adding new classes inheriting from the Layer class. These new layers have then to be assigned a short name and registered in the LayerFactory.

It is also possible to define new protocol ports. To do so, it is necessary to implement new classes inheriting from the ProtocolPort class. These new ports have then to be assigned a short name and registered in the ProtocolPortFactory.

Furthermore, it is also possible to extend AC primitives. This requires to enrich IManagementLayers and IManagementTA interfaces and to implement new functionalities in the Management class.

7.2.4 Adapter Control primitives

The following adapter control primitives are used to control the dynamic configuration of the various layers:

Table 1: Adapter Control primitives

Adapter Control Primitive	Description
startBeaconing	Requests Test Adapter to start sending periodic beacons for the current component
stopBeaconing	Requests Test Adapter to stop sending periodic beacons for the current component
startEnqueueingBeacons	Requests Test Adapter to start enqueueing beacon messages on the current component GN port
stopEnqueueingBeacons	Requests Test Adapter to stop enqueueing beacon messages on the current component GN port
startMultipleBeaconing	Requests Test Adapter to start simulating neighbour presence by sending multiple periodic beacons for the current component
stopMultipleBeaconing	Requests Test Adapter to stop simulating neighbour presence
getLongPositionVector	Gets the long position vector of a neighbour given its GN Address

7.3 Platform Adapter

All TTCN-3 commercial tools provide generic Platform Adapter implementations for managing TTCN-3 timers. These implementations are well tested and usually accurate enough for most usages. In the case of ITS protocols, e.g. DENM re-broadcasting, GN beacon interval, etc., the protocol timer value is in the order of magnitude of hundreds of milliseconds. This order of magnitude can be handled well with the built in test system timers. As a consequence, no specific development is required for this component.

7.4 Upper Tester

The upper tester is used to interact with the upper interface of the implementation under test (IUT). It is typically implemented as an upper tester module executing in the test adapter and as a small module executing on the SUT and acts as an upper layer for the IUT, as shown in figure 9. It is particularly useful for:

- Triggering events in SUT.
- Triggering messages.

- Checking that payload are transmitted correctly to upper layers.

The communication between the two upper tester modules is performed in accordance with the upper tester message format described in annex C.

As it interacts with potentially proprietary APIs, it is usually the responsibility of IUT vendors to implement module located within SUT.

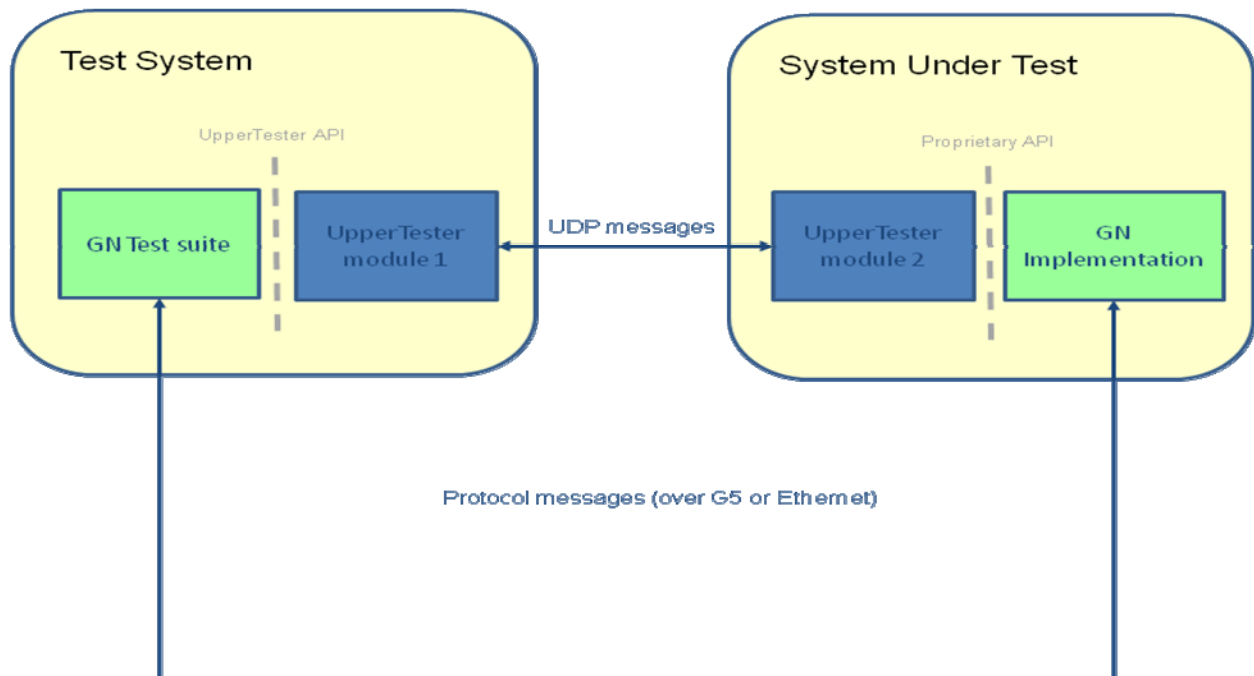


Figure 9: Upper Tester architecture

- This upper tester module implements the upper tester message based API described in annex C.

Annex A: Codecs Source Code

The applicable software tag of the codec source code is:

<http://forge.etsi.org/websvn/listing.php?repname=ITS.ITS&path=/tags/v1.2.1/>

Annex B:

Test Adapter Source Code

The applicable software tag of the test adapter source code is:

<http://forge.etsi.org/websvn/listing.php?repname=ITS.ITS&path=/tags/v1.2.1/>

Annex C:

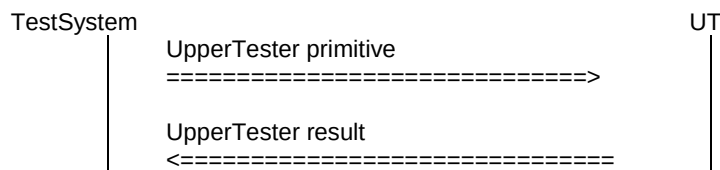
Upper Tester Message Format

The messages defined in the present annex are exchanged between the Test System and Upper Tester using a UDP connection.

All integer values are encoded in big-endian byte order (most significant byte first).

Two different message exchanges can occur.

- The first communication exchange is initiated by the test system and consists in a request – response exchange as described below. The UpperTester result message is specific to each primitive and may be used to indicate the success of the request or to report some values.



In this case the UDP destination port of the response is identical to the UDP source port of the request. When receiving UtInitialize primitive from Test System, the UDP source port of this request is saved as 'defaultUTPort' and used for unsolicited indications.

- The second communication exchange is initiated by the Upper Tester. It consists in unsolicited indications sent each time a packet is transmitted to upper layers, as described below. The Test System never replies to such messages (one way communication)



In this case, the UDP destination port of the indication is set to the 'defaultUTPort', which corresponds to the UDP source port of the UTInitialize request.

Format of UtResult:

0	1
0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7
MessageType = 0x24	Result

Name	Length	Value
MessageType	1 byte	0x24
Result	1 byte	0x00: Failure 0x01: Success

C.1 Common Upper Tester Primitives

C.1.1 UtInitialize

This message is used to request initialization of IUT implementation.

Request (UtInitialize TS → UT):

```

0
0 1 2 3 4 5 6 7
Message Type = 0x00

```

Name	Length	Value
Message Type	1 byte	0x00

Response (UtInitializeResult UT → TS):

```

0                               1
0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
Message Type = 0x01 Result

```

Name	Length	Value
Message Type	1 byte	0x01
Result	1 byte	0x00: Failure 0x01: Success

NOTE: The notation "TS → UT" and "UT → TS" is used in this clause and all sub sequent clauses, and signifies "from TS to UT" and "from UT to TS".

C.1.2 ChangePosition

This message is used to change the position of the ITS station. The latitude, longitude and elevation parameters are relative to the current position of IUT. They are NOT absolute position.

Request (UtChangePosition TS → UT):

```

0                               1                               2                               3
0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
Message Type = 0x02 DeltaLatitude
... DeltaLatitude
... DeltaElevation

```

Name	Length	Value
Message Type	1 byte	0x02
DeltaLatitude	4 bytes	Latitude offset (multiples of 0,1 microdegree)
DeltaLongitude	4 bytes	Longitude offset (multiples of 0,1 microdegree)
DeltaElevation	1 byte	Elevation offset (meter)

Response (UtChangePositionResult UT → TS):

```

0                               1
0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
Message Type = 0x03 Result

```

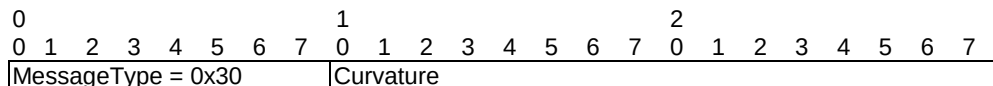
Name	Length	Value
Message Type	1 byte	0x03
Result	1 byte	0x00: Failure 0x01: Success

C.2 CAM Upper Tester Primitives

C.2.1 ChangeCurvature

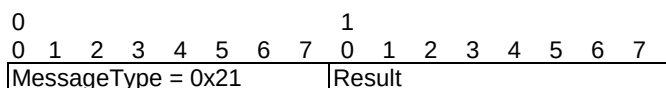
This message is used to set the curvature of the ITS station.

Request (UtCamTrigger_changeCurvature TS → UT):



Name	Length	Value
MessageType	1 byte	0x30
Curvature	2 bytes	Signed integer. Curvature change from -30 000 to 30 001

Response (UtCamTriggerResult UT → TS):



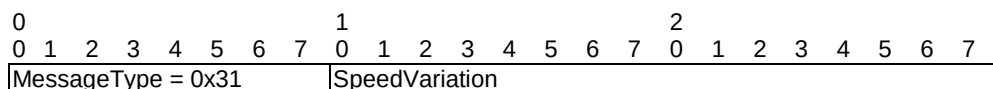
Name	Length	Value
MessageType	1 byte	0x21
Result	1 byte	0x00: Failure 0x01: Success

C.2.2 ChangeSpeed

This message is used to change the speed of the ITS station. The vehicle speed is increased by the value of 'SpeedVariation' field.

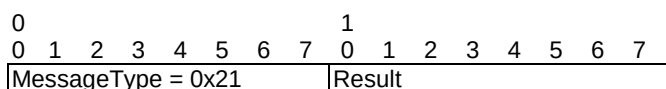
For instance, if the current speed of the ITS station is 10 m/s and received SpeedVariation is +300, then the new vehicle speed will be $10 + 0,01 \times 300 = 13$ m/s.

Request (UtCamTrigger_changeSpeed TS → UT):



Name	Length	Value
MessageType	1 byte	0x31
SpeedVariation	2 bytes	Signed integer. Speed variation in units of m/s

Response (UtCamTriggerResult UT → TS):



Name	Length	Value
MessageType	1 byte	0x21
Result	1 byte	0x00: Failure 0x01: Success

C.2.3 SetAccelerationControlStatus

brakePedalActive (0),

gasPedalActive (1),

emergencyBrakeActive (2),

collisionWarningActive (3),

accActive (4),

cruiseControl (5),

speedLimiterActive (6)

This message is used to set acceleration control status of the ITS station.

Request (UtCamTrigger_setAccelerationControlStatus TS → UT):

0	1
0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7
MessageType = 0x32	B G E C A C L X

Name	Length	Value
MessageType	1 byte	0x33
B	1 bit	0: brake pedal inactive 1: brake pedal active
G	1 bit	0: gas pedal inactive 1: gas pedal active
E	1 bit	0: emergency brake inactive 1: emergency brake active
C	1 bit	0: collision warning inactive 1: collision warning active
A	1 bit	0: ACC inactive 1: ACC active
CC	1 bit	0: cruise control inactive 1: cruise control active
L	1 bit	0: speed limiter inactive 1: speed limiter active
X	1 bit	Reserved

Response (UtChangePositionResult UT → TS):

0	1
0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7
MessageType = 0x21	Result

Name	Length	Value
MessageType	1 byte	0x21
Result	1 byte	0x00: Failure 0x01: Success

This message is used to set exterior lights status of the ITS station.

0								1							
0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
MessageType = 0x33								L	H	L	R	D	R	F	P
								B	B	T	T				

Name	Length	Value
MessageType	1 byte	0x33
LB	1 bit	0: low beam headlights off 1: low beam headlights on
HB	1 bit	0: high beam off 1: high beam headlights on
LT	1 bit	0: left turn signal off 1: left turn signal on
RT	1 bit	0: right turn signal off 1: right turn signal on
D	1 bit	0: daytime running lights off 1: daytime running lights on
R	1 bit	0: reverse light off 1: reverse lights on
F	1 bit	0: fog light off 1: fog light on
P	1 bit	0: parking lights off 1: parking lights on

0								1							
0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
MessageType = 0x21								Result							

Name	Length	Value
MessageType	1 byte	0x32
Result	1 byte	0x00: Failure 0x01: Success

C.2.5 ChangeHeading

This message is used to change the heading of the ITS station. The new heading of the ITS station is reflect Direction parameter.

Request (UtCamTrigger_changeHeading TS → UT):

0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
MessageType = 0x34								Direction															

Name	Length	Value
MessageType	1 byte	0x34
Direction	2 bytes	Integer value from 0 to 3 600 0: North 900: East 1 800: South 2 700: West 3 600: Unavailable

Response (UtCamTriggerResult UT → TS):

0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
MessageType = 0x21								Result							

Name	Length	Value
MessageType	1 byte	0x21
Result	1 byte	0x00: Failure 0x01: Success

C.2.6 SetDriveDirection

This message is used to change the direction of the ITS station.

Request (UtCamTrigger_setDriveDirection TS → UT):

0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
MessageType = 0x35								Direction							

Name	Length	Value
MessageType	1 byte	0x35
Direction	1 byte	0x00: Forward 0x01: Backward 0x02: Unavailable

Response (UtCamTriggerResult UT → TS):

0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
MessageType = 0x21								Result							

Name	Length	Value
MessageType	1 byte	0x21
Result	1 byte	0x00: Failure 0x01: Success

C.2.7 ChangeYawRate

This message is used to change the yaw rate of the ITS station.

Request (UtCamTrigger_changeYawRate TS → UT):

0								1								2							
0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
Message Type = 0x36								YawRate															

Name	Length	Value
Message Type	1 byte	0x36
YawRate	2 bytes	Signed integer. Curvature range from -32 767 to -32 767

Response (UtCamTriggerResult UT → TS):

0	1
0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7
Message Type = 0x21	Result

Name	Length	Value
Message Type	1 byte	0x21
Result	1 byte	0x00: Failure 0x01: Success

C.2.8 CamEventIndication

This message is used to indicate reception of CAM information by IUT.

Indication (UtCamEventInd UT → TS):

0	1	2	3
0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7
Message Type = 0x23	CamPduLength	CamPdu	:
...			

Name	Length	Value
Message Type	1 byte	0x23
CamPduLength	2 bytes	Length of 'CamPdu' field
CamPdu	Variable	Received CAM

C.2.9 SetStationType

unknown(0),

pedestrian(1),

cyclist(2),

moped(3),

motorcycle(4),

passengerCar(5),

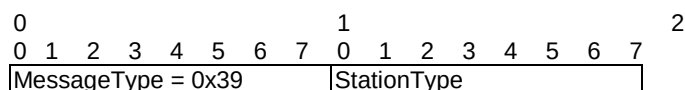
bus(6),

lightTruck(7),

heavyTruck(8),
 trailer(9),
 specialVehicles(10),
 tram(11),
 roadSideUnit(15)

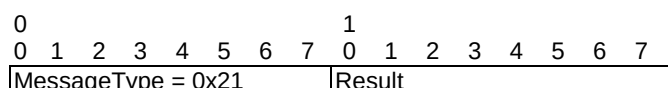
This message is used to change the type of the ITS station.

Request (UtCamTrigger_setStationType TS → UT):



Name	Length	Value
MessageType	1 byte	0x39
StationType	1 byte	Unsigned char range from 0 to 15

Response (UtCamTriggerResult UT → TS):



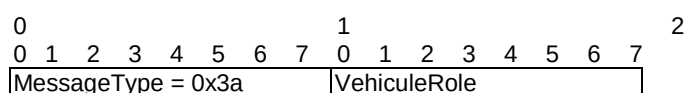
Name	Length	Value
MessageType	1 byte	0x21
Result	1 byte	0x00: Failure 0x01: Success

C.2.10 SetVehicleRole

default(0),
 publicTransport(1),
 specialTransport(2),
 dangerousGoods(3),
 roadWork(4),
 rescue(5),
 emergency(6),
 safetyCar(7)

This message is used to change the vehicle role of the ITS station.

Request (UtCamTrigger_setVehicleRole TS → UT):



Name	Length	Value
MessageType	1 byte	0x3a
VehiculeRole	1 byte	Unsigned char range from 0 to 7

Response (UtCamTriggerResult UT → TS):

0	1
0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7
MessageType = 0x21	Result

Name	Length	Value
MessageType	1 byte	0x21
Result	1 byte	0x00: Failure 0x01: Success

C.2.11 SetEmbarkationStatus

This message is used to indicate whether the passenger embarkation is ongoing.

Request (UtCamTrigger_setEmbarkationStatus TS → UT):

0	1	2
0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7
MessageType = 0x3b	EmbarkationStatus	

Name	Length	Value
MessageType	1 byte	0x3b
EmbarkationStatus	1 byte	Unsigned char. Value is 0 for false and value is 255 for true

Response (UtCamTriggerResult UT → TS):

0	1
0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7
MessageType = 0x21	Result

Name	Length	Value
MessageType	1 byte	0x21
Result	1 byte	0x00: Failure 0x01: Success

C.2.12 SetPtActivation

This message is used to control traffic lights, barriers, etc.

Request (UtCamTrigger_setPtActivation TS → UT):

0	1	2	
0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7	
MessageType = 0x3c	PtActivationType	PtActivationDataLength	PtActivatioData
...			

Name	Length	Value
MessageType	1 byte	0x3c
PtActivationType	1 byte	Unsigned char range from 0 to 255
PtActivationDataLength	1 byte	Unsigned char range from 0 to 20
PtActivatioData	Variable	Unsigned char range from 0 bytes to 20 bytes

Response (UtCamTriggerResult UT → TS):

0	1
0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7
MessageType = 0x21	Result

Name	Length	Value
MessageType	1 byte	0x21
Result	1 byte	0x00: Failure 0x01: Success

C.2.13 SetDangerousGoods

explosives1(0),
 explosives2(1),
 explosives3(2),
 explosives4(3),
 explosives5(4),
 explosives6(5),
 flammableGases(6),
 nonFlammableGases(7),
 toxicGases(8),
 flammableLiquids(9),
 flammableSolids(10),
 substancesLiableToSpontaneousCombustion(11),
 substancesEmittingFlammableGasesUponContactWithWater(12),
 oxidizingSubstances(13),
 organicPeroxides(14),
 toxicSubstances(15),
 infectiousSubstances(16),
 radioactiveMaterial(17),
 corrosiveSubstances(18),
 miscellaneousDangerousSubstances(19)

This message is used to set the dangerous good property of the ITS station.

Request (UtCamTrigger_setDangerousGoods TS → UT):

0								1								2							
0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7								
MessageType = 0x3d								DangerousGood															

Name	Length	Value
Message Type	1 byte	0x3d
DangerousGood	1 byte	Unsigned char range from 0 to 19

Response (UtCamTriggerResult UT → TS):

0	1
0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7
Message Type = 0x21	Result

Name	Length	Value
Message Type	1 byte	0x21
Result	1 byte	0x00: Failure 0x01: Success

C.2.14 SetDangerousGoodsExtended

This message is used to change the vehicle role of the ITS station.

Request (UtCamTrigger_setDangerousGoodsExt TS → UT):

0								1								2							
0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
MessageType = 0x3e								DangerousGoodExt															

Name	Length	Value
Message Type	1 byte	0x3e
DangerousGoodExt	1 byte	Unsigned char range from 0 to 255

Response (UtCamTriggerResult UT → TS):

0	1
0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7
Message Type = 0x21	Result

Name	Length	Value
Message Type	1 byte	0x21
Result	1 byte	0x00: Failure 0x01: Success

C.2.15 SetLightBarSiren

This message is used to set light and siren bar status of the ITS station.

Request (UtCamTrigger_setLightBarSiren TS → UT):

0	1
0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7
Message Type = 0x3f	L S
	B

Name	Length	Value
Message Type	1 byte	0x3f
LB	1 bit	0: Light bar is not activated 1: Light bar is activated
S	1 bit	0: Siren is off 1: Siren is on

Response (UtCamTriggerResult UT → TS):

0	1
0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7
Message Type = 0x21	Result

C.3 DENM Upper Tester Primitives

C.3.1 GenerateDenmEvent

This message is used to create a new DENM event.

Request (UtDenmTrigger TS → UT):

0	1	2	3
0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7
Message Type = 0x10	V R X	I X	Detection Time
...			
Validity Duration		Info Quality	Repetition Duration
...			Cause
Sub Cause	Relevance Distance	Traffic Direction	Transmission Interval
...	KAF		

Name	Length	Value
MessageType	1 byte	0x10
V	1 bit	0: validityDuration to be ignored 1: validityDuration to be used
R	1 bit	0: RepetitionDuration to be ignored 1: RepetitionDuration to be used
X	3 bits	reserved
I	1 bit	0: TransmissionInterval to be ignored 1: TransmissionInterval to be used
X	2 bits	reserved
DetectionTime	6 bytes	Unsigned integer. From 0 to 3 153 600 000 000
ValidityDuration	3 bytes	Unsigned integer. From 0 s to 86 400 s
RepetitionDuration	3 bytes	Unsigned integer. From 0 s to 86 400 s
InfoQuality	1 byte	0x00: Unavailable 0x01: Lowest ... 0x07: Highest
Cause	1 byte	Event cause ID
Subcause	1 byte	Event sub-cause ID
RelevanceDistance	1 byte	0x00: less than 50 m 0x01: less than 100 m 0x02: less than 200 m 0x03: less than 500 m 0x04: less than 1 000 m 0x05: less than 5 km 0x06: less than 10 km 0x07: greater than 10 km
TrafficDirection	1 byte	0x00: unavailable 0x01: upstream traffic 0x02: downstream traffic 0x03: all traffic directions
TransmissionInterval	2 bytes	From 1 ms to 10 000 ms
KAF	1 byte	0x00: No Keep-Alive Forwarding 0x01: Keep-Alive Forwarding

Response (UtDenmTriggerResult UT → TS):

0	1	2	3
0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7
MessageType = 0x11	Result	StationId	SequenceNo
...			

Name	Length	Value
MessageType	1 byte	0x11
Result	1 byte	Operation result
StationId	4 bytes	Station ID
SequenceNo	2 bytes	Event sequence number

C.3.2 UpdateDenmEvent

This message is used to update expiration time of an existing DENM event.

Request (UtDenmUpdate TS → UT):

0								1								2								3															
0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7								
MessageType = 0x12								V	S	D	T	C	I	X	StationId																								
...								SequenceNo																															
DetectionTime																																							
...								ValidityDuration																															
...								InfoQuality																Cause								SubCause							
RelevanceDistance								TrafficDirection																TransmissionInterval															
KAF																																							

Name	Length	Value
MessageType	1 byte	0x12
V	1 bit	0: validityDuration to be ignored 1: validityDuration to be used
S	1 bit	0: InfoQuality, CauseCaude SubCauseCode to be ignored 1: InfoQuality, CauseCaude SubCauseCode to be used
D	1 bit	0: relevanceDistance to be ignored 1: relevanceDistance to be used
T	1 bit	0: TrafficDirection to be ignored 1: TrafficDirection to be used
C	1 bit	0: TrafficClass to be ignored 1: TrafficClass to be used
I	1 bit	0: TransmissionInterval to be ignored 1: TransmissionInterval to be used
X	2 bits	reserved
StationId	4 bytes	Original event's station ID
SequenceNo	2 bytes	Original event's sequence number
DetectionTime	6 bytes	Unsigned integer. From 0 to 3 153 600 000 000
ValidityDuration	3 bytes	Unsigned integer. From 0 s to 86 400 s
InfoQuality	1 byte	0x00: Unavailable 0x01: Lowest ... 0x07: Highest
Cause	1 byte	Event cause ID
Subcause	1 byte	Event sub-cause ID
RelevanceDistance	1 byte	0x00: less than 50 m 0x01: less than 100 m 0x02: less than 200 m 0x03: less than 500 m 0x04: less than 1000 m 0x05: less than 5 km 0x06: less than 10 km 0x07: greater than 10 km
TrafficDirection	1 byte	0x00: unavailable 0x01: upstream traffic 0x02: downstream traffic 0x03: all traffic directions
TransmissionInterval	2 bytes	From 1 ms to 10 000 ms
KAF	1 byte	0x00: No Keep-Alive Forwarding 0x01: Keep-Alive Forwarding

Response (UtDenmUpdateResult UT → TS):

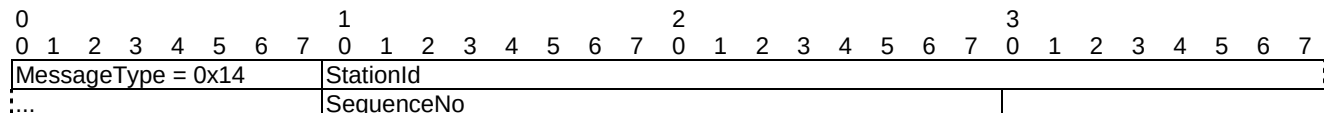
0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
MessageType = 0x13								Result								StationId							
...																SequenceNo							

Name	Length	Value
MessageType	1 byte	0x13
Result	1 byte	Operation result
StationId	4 bytes	Station ID
SequenceNo	2 bytes	Event sequence number

C.3.3 TerminateDenmEvent

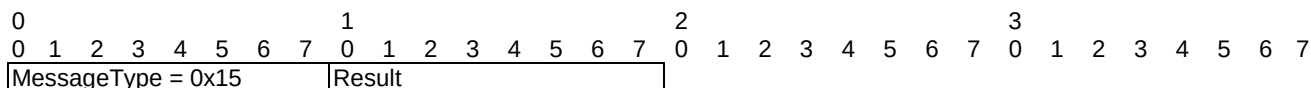
This message is used to terminate an existing DENM event.

Request (UtDenmTermination TS → UT):



Name	Length	Value
MessageType	1 byte	0x14
StationId	4 bytes	Original Station ID
SequenceNo	2 bytes	Event sequence number

Response (UtDenmTerminationResult UT → TS):

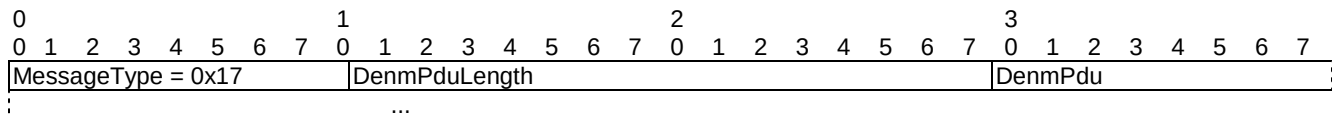


Name	Length	Value
MessageType	1 byte	0x15
Result	1 byte	0x00: Failure 0x01: Success

C.3.4 DenmEventIndication

This message is used to indicate reception of DENM information by IUT.

Indication (UtDenmEventInd UT → TS):



Name	Length	Value
MessageType	1 byte	0x17
DenmPduLength	2 bytes	Length of 'DenmPdu' field
DenmPdu	Variable	Received DENM

C.4 GeoNetworking Upper Tester Primitives

C.4.1 GenerateGeoUnicast

This message is used to trigger a GeoUnicast message.

Request (UtGnTrigger_geoUnicast TS → UT):

0								1								2								3							
0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
MessageType = 0x50								DstGnAddress																							
...																															
								Lifetime																PayloadLength							
								Payload																							

Name	Length	Value
MessageType	1 byte	0x50
DstGnAddr	8 bytes	Destination GN Address
Lifetime	2 bytes	Packet lifetime in milliseconds
PayloadLength	2 bytes	Length of 'Payload' field
Payload	Variable	Packet's final payload

Response (UtGnTriggerResult UT → TS):

0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
MessageType = 0x41								Result							

Name	Length	Value
MessageType	1 byte	0x41
Result	1 byte	0x00: Failure 0x01: Success

C.4.2 GenerateGeoBroadcast

This message is used to trigger a GeoBroadcast message.

Request (UtGnTrigger_geoBroadcast TS → UT):

0								1								2								3															
0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7								
MessageType = 0x51								Shape								Lifetime																							
Latitude																																							
Longitude																																							
DistanceA																DistanceB																							
Angle																PayloadLength																							
Payload																																							

Name	Length	Value
MessageType	1 byte	0x51
Shape	1 byte	0: Circle 1: Rectangle 2: Ellipse
Lifetime	2 bytes	Packet lifetime in milliseconds
Latitude	4 bytes	Destination area latitude (1/10 degrees)
Longitude	4 bytes	Destination area longitude (1/10 degrees)
DistanceA	2 bytes	Destination area distance A
DistanceB	2 bytes	Destination area distance B
Angle	2 bytes	Destination area angle
PayloadLength	2 bytes	Length of 'Payload' field
Payload	Variable	Packet's final payload

Response (UtGnTriggerResult UT → TS):

0	1
0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7
MessageType = 0x41	Result

Name	Length	Value
MessageType	1 byte	0x41
Result	1 byte	0x00: Failure 0x01: Success

C.4.3 GenerateGeoAnycast

This message is used to trigger a GeoAnycast message.

Request (UtGnTrigger_geoAnycast TS → UT):

0	1	2	3
0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7
MessageType = 0x52	Shape	Lifetime	
Latitude			
Longitude			
DistanceA		DistanceB	
Angle		PayloadLength	
Payload			

Name	Length	Value
MessageType	1 byte	0x52
Shape	1 byte	0: Circle 1: Rectangle 2: Ellipse
Lifetime	2 bytes	Packet lifetime in milliseconds
Latitude	4 bytes	Destination area latitude (1/10 degrees)
Longitude	4 bytes	Destination area longitude (1/10 degrees)
DistanceA	2 bytes	Destination area distance A
DistanceB	2 bytes	Destination area distance B
Angle	2 bytes	Destination area angle
PayloadLength	2 bytes	Length of 'Payload' field
Payload	Variable	Packet's final payload

Response (UtGnTriggerResult UT → TS):

0	1
0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7
MessageType = 0x41	Result

Name	Length	Value
MessageType	1 byte	0x41
Result	1 byte	0x00: Failure 0x01: Success

C.4.4 GenerateSHB

This message is used to trigger a SHB message.

Request (UtGnTrigger_shb TS → UT):

0	1	2	3
0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7
MessageType = 0x53	PayloadLength	Payload	:

Name	Length	Value
MessageType	1 byte	0x53
PayloadLength	2 bytes	Length of 'Payload' field
Payload	Variable	Packet's final payload

Response (UtGnTriggerResult UT → TS):

0	1
0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7
MessageType = 0x41	Result

Name	Length	Value
MessageType	1 byte	0x41
Result	1 byte	0x00: Failure 0x01: Success

C.4.5 GenerateTSB

This message is used to trigger a TSB message.

Request (UtGnTrigger_tsb TS → UT):

0	1	2	3
0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7
MessageType = 0x54	NbHops	PayloadLength	:
Payload			:

Name	Length	Value
MessageType	1 byte	0x54
NbHops	1 byte	Number of hops
PayloadLength	2 bytes	Length of 'Payload' field
Payload	Variable	Packet's final payload

Response (UtGnTriggerResult UT → TS):

0	1
0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7
MessageType = 0x41	Result

Name	Length	Value
MessageType	1 byte	0x41
Result	1 byte	0x00: Failure 0x01: Success

C.4.6 GnEventIndication

This message is used to check whether payload contained in GeoNetworking PDU has been transmitted to upper layer (CAM/DENM/IPv6).

Indication (UtGnEventInd UT → TS):

0	1	2	3
0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7
MessageType = 0x17	PacketLength	Packet	:
:	...	:	:

Name	Length	Value
MessageType	1 byte	0x17
PacketLength	2 bytes	Length of 'Packet' field
Packet	Variable	Packet's final payload

C.5 IPv6OverGeoNetworking Upper Tester Primitives

C.5.1 SendIPv6Message

This message is used to trigger the sending of an IPv6 message on a specified network interface.

Request (UtGn6Trigger TS → UT):

0	1	2	3
0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7
MessageType = 0x80	InterfaceLength	InterfaceName	
SrcMacAddress		DestMacAddress	
PacketLength		IPv6Packet	

Name	Length	Value
MessageType	1 byte	0x80
InterfaceLength	1 byte	Length of "InterfaceName" field
InterfaceName	InterfaceLength × 1 byte	Name of the interface on which to send the IPv6 packet
SrcMacAddress	6 bytes	Source MAC address
DestMacAddress	6 bytes	Destination MAC address
PacketLength	2 bytes	Length of the "IPv6Packet" field
IPv6Packet	Variable	IPv6 packet to be sent

Response (UtGn6TriggerResult UT → TS):

0	1
0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7
MessageType = 0x81	Result

Name	Length	Value
MessageType	1 byte	0x81
Result	1 byte	0x00: Failure 0x01: Success

C.5.2 GetInterfaceInfos

This message is used by the Test System to retrieve the configuration of network interfaces on IUT.

Request (UtGn6GetInterfaceInfo TS → UT):

0
0 1 2 3 4 5 6 7
MessageType = 0x84

Name	Length	Value
MessageType	1 byte	0x84

Response (UtGn6GetInterfaceInfoResult UT → TS):

0	1	2	3
0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7
MessageType = 0x85	InterfaceCount	InterfaceLength[0]	InterfaceName[1]
...	AddressCount[0]	Addresses[0][0]	
...			
...			
...		Addresses[0][1]	
...	...	InterfaceLength[1]	InterfaceName[1]
...	...		

Name	Length	Value
MessageType	1 byte	0x85
InterfaceCount	1 byte	Number of interface descriptors
Interface Descriptor	InterfaceLength	1 byte
	InterfaceName	InterfaceLength × 1 byte
	AddressCount	1 byte
	Addresses	AddressCount × 16 bytes
		Length of "InterfaceName" field
		Name of the interface
		Number of configured IPv6 address on the interface
		IPv6 addresses configured on interface

C.5.3 Gn6EventIndication

This message is used to check whether payload contained in GeoNetworking PDU has been transmitted to upper layer (CAM/DENM/IPv6).

Indication (UtGnEventInd UT → TS):

0	1	2	3
0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7
MessageType = 0x83	InterfaceLength	InterfaceName	...
PacketLength		IPv6Packet	

Name	Length	Value
MessageType	1 byte	0x83
InterfaceLength	1 byte	Length of "InterfaceName" field
InterfaceName	InterfaceLength × 1 byte	Name of the interface on the IPv6 packet has been received
PacketLength	2 bytes	Length of 'IPv6Packet' field
IPv6Packet	Variable	Received IPv6 packet

C.6 BTP Upper Tester Primitives

C.6.1 GenerateBtpA

This message is used to trigger a BTP-A message.

Request (UtBtpTrigger_A TS → UT):

0	1	2	3
0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7
MessageType = 0x70	DestPort	SrcPort	:
...			

Name	Length	Value
MessageType	1 byte	0x70
DestPort	2 bytes	Destination port
SrcPort	2 bytes	Source port

Response (UtBtpTriggerResult UT → TS):

0	1
0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7
MessageType = 0x61	Result

Name	Length	Value
MessageType	1 byte	0x61
Result	1 byte	0x00: Failure 0x01: Success

C.6.2 GenerateBtpB

This message is used to trigger a BTP-B message.

Request (UtBtpTrigger_B TS → UT):

0	1	2	3
0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7
MessageType = 0x71	DestPort	DestPortInfo	:
...			

Name	Length	Value
MessageType	1 byte	0x71
DestPort	2 bytes	Destination port
DestPortInfo	2 bytes	Destination port info

Response (UtBtpTriggerResult UT → TS):

0	1
0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7
MessageType = 0x61	Result

Name	Length	Value
MessageType	1 byte	0x61
Result	1 byte	0x00: Failure 0x01: Success

C.6.3 BtpEventIndication

This message is used to check whether payload contained in BTP PDU has been transmitted to upper layer.

Indication (UtBtpEventInd UT → TS):

0	1	2	3
0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7
MessageType = 0x63	PacketLength	Packet	:
:	...		

Name	Length	Value
MessageType	1 byte	0x63
PacketLength	2 bytes	Length of 'Packet' field
Packet	Variable	Packet's final payload

Annex D:

Example of Test Platform implementation

The test platform used by ETSI STF424 for validating ITS conformance test suites has been developed using the following tools and components:

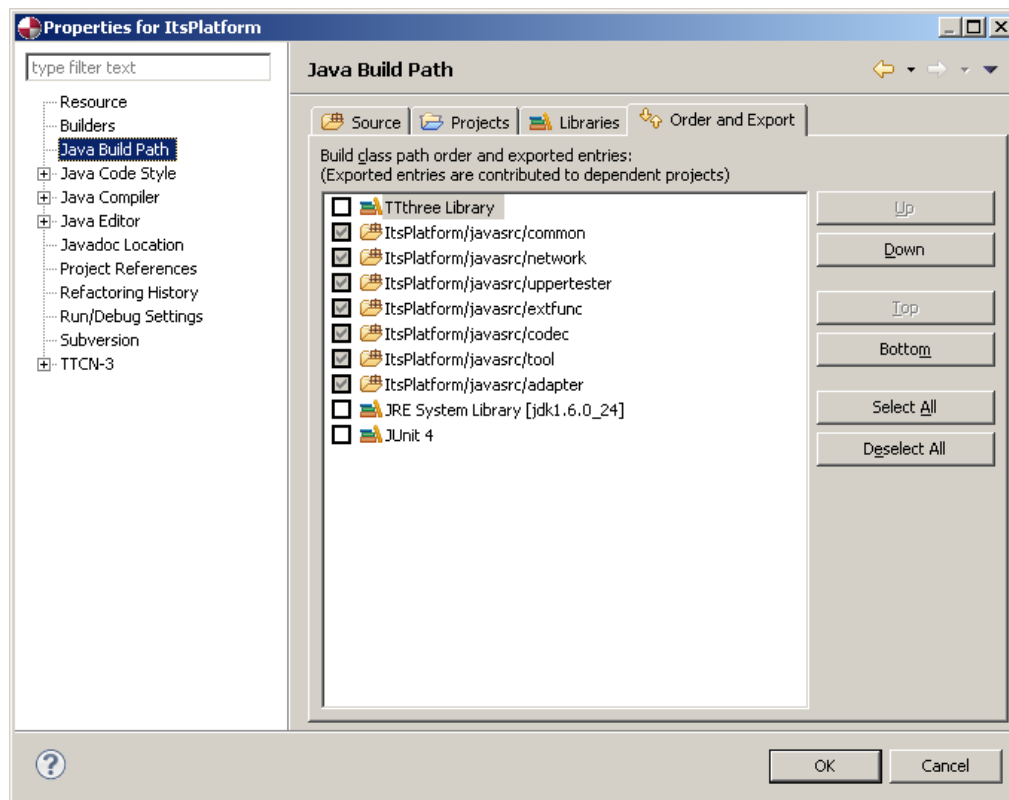
- Standard PC equipped with two Ethernet network cards (It is possible to use the same single network card for both tasks, but it is less practical and gives less flexibility).
One network card (Mac address: 00-A0-24-AD-56-FF) is used to communicate with G5 device.
The second one (Mac address: 00-50-56-C0-00-08) is used to establish upper tester link with SUT and is configured with IP address 192.168.56.1/24.
- Windows™ XP Professional operating system (32 bits)
No special requirement concerning operating system. Theoretically, the platform can be used on Linux based operating systems, as it is OS independent.
- Testing Technologies TTworkbench Basic v13 with ASN.1 plugins
ASN.1 plugins are necessary for CAM and DENM codecs. Any other TTCN-3 test tool would be suitable with minimum adaptation as the test platform is tool independent.
- Java™ JDK 1.6.0-24
All the software used in the test platform have been developed using Java™ language.
- JnetPcap 1.3.0
This library is used for capturing and injecting raw Ethernet packets. It is a direct dependency of EthernetLayer module. For easy setup the jnetpcap.dll file has to be installed in C:\Windows\System32\ folder and the jnetpcap.jar has to be installed in C:\Windows\Sun\Java\lib\ext\ folder or equivalent. By doing this, no specific setting will be required to include JnetPcap library when building Test Adapter.
- Cohda Wireless™ MK2
This device provides G5 connectivity to the test platform. This device features a G5 radio interface used to communicate with SUT and Ethernet interface that is connect to the test platform PC in order to transfer G5 packets to be sent and received via the radio interface.

Before running successfully any test, a certain number of settings have to be verified in TTworkbench:

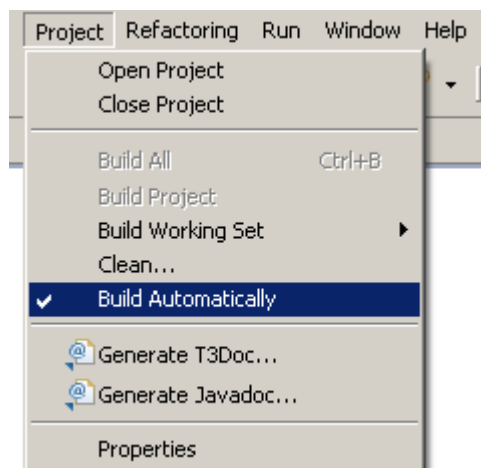
- Project has to be set for using Java™ JDK 1.6.0 as indicating in the following picture. Please note that JnetPcap should automatically appear in the library list:



- Test Adapter and Codecs source folders have to be declared in project's Java™ Build Path:

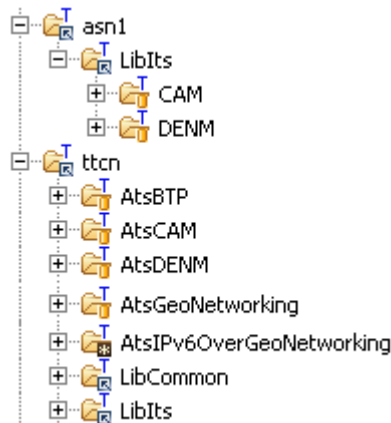


- Test Adapter and Codecs will then be automatically compiled if "Build automatically" option is set:



- Alternatively, Test Adapter and Codecs precompiled libraries have to be referenced as external libraries

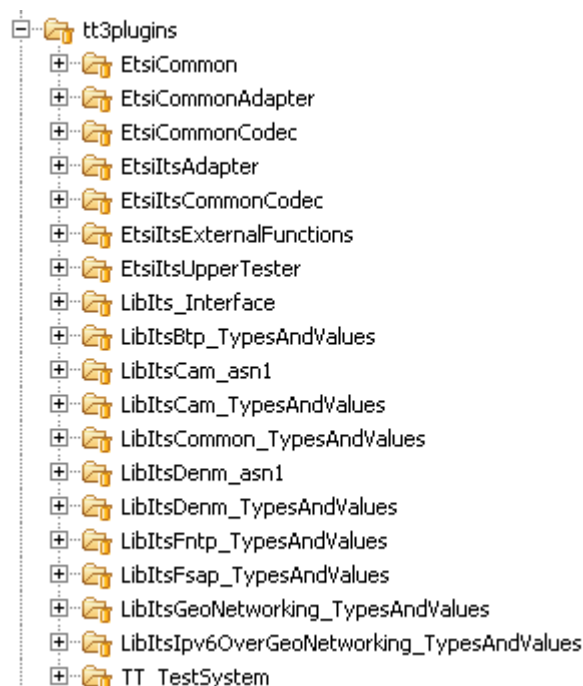
- TTCN-3 test suites and ASN.1 definitions have to be copied to the project and declared as TTCN-3 source folders:



- The test suites need to be compiled using the "Rebuild All" button:



- TT3 plugins have to be configured using the provided xml files:



- Test adapter parameters have to be adjusted in taconfig.xml file:

```
<parameter id="taParams">
  <parameter id="DEBUG_ENABLED" value="ALL"/>
  <parameter id="camPort" value="BTP/GN/ETH"/>
  <parameter id="denmPort" value="BTP/GN/ETH"/>
  <parameter id="btpPort" value="GN/ETH"/>
  <parameter id="geoNetworkingPort" value="ETH"/>
  <parameter id="ipv6OverGeoNetworkingPort" value="Debug"/>
  <parameter id="CamUpperTester" value="Operator"/>
  <parameter id="CamUpperTesterSettings" value=""/>
  <parameter id="DenmUpperTester" value="Operator"/>
  <parameter id="DenmUpperTesterSettings" value=""/>
  <parameter id="BtpUpperTester" value="Operator"/>
  <parameter id="BtpUpperTesterSettings" value=""/>
  <parameter id="GnUpperTester" value="Generic"/>
  <parameter id="GnUpperTesterSettings" value=""/>
</parameter>
```

```

value="NwtaTrigger:192.168.56.10:1600:1601"/>
<parameter id="LocalEthernetMAC" value="00A024AD56FF"/>
<parameter id="IutEthernetTypeValue" value="0x0707"/>
<parameter id="LinkLayer_MTC" value="BABEBABE0000"/>
<parameter id="LinkLayer_NodeA" value="BABEBABE0001"/>
<parameter id="LinkLayer_NodeB" value="BABEBABE0002"/>
<parameter id="LinkLayer_NodeC" value="BABEBABE0003"/>
<parameter id="LinkLayer_NodeD" value="BABEBABE0004"/>
<parameter id="Gn6RemoteAdapterIp" value="192.168.56.10"/>
<parameter id="Gn6RemoteAdapterPort" value="42000"/>
</parameter>

```

Table D.1 summarizes the authorized values for these parameters.

Table D.1: Test Adapter Parameters

Parameter	Description	Allowed values
DEBUG_ENABLED	Indicates whether Codecs and Test Adapter produce debugging logs	ALL, NONE, OFF
camPort	Defines the lower stack of CamPort	Any combination of valid layer identifier separated by "/" symbol: • ETH • BTP • GN • UdpIp • Debug (pseudo layer that dumps packet to console) • Loopback (pseudo layer that reinjects the packets)
denmPort	Defines the lower stack of CamPort	
btpPort	Defines the lower stack of CamPort	
geoNetworkingPort	Defines the lower stack of CamPort	
ipv6OverGeoNetworkingPort	Defines the lower stack of CamPort	
CamUpperTester DenmUpperTester BtpUpperTester GnUpperTester	Selects the type of Upper tester to be used for each test suite	Operator, Yes, Generic
CamUpperTesterSettings DenmUpperTesterSettings BtpUpperTesterSettings GnUpperTesterSettings	Defines Upper Tester specific settings like remote IP addresses, UDP ports, etc.	Upper tester specific
LocalEthernetMAC	MAC Address of the Ethernet card used to communicate with G5 equipment	Hexstring representation of Mac Address without separator
IutEthernetTypeValue	Ethertype value to be used for sending and capturing packets	Integer 0 to 65 535. Should be 0x0707
LinkLayer_MTC LinkLayer_NodeA LinkLayer_NodeB LinkLayer_NodeC LinkLayer_NodeD	MAC addresses used by simulated ITS nodes	Hexstring representation of Mac Address without separator
Gn6RemoteAdapterIp	IP Address of GN6 Remote Adapter	Standard IP address notation
Gn6RemoteAdapterPort	UDP port of GN6 Remote Adapter	Integer 0 to 65 535.

Annex E:

Complete Test Adapter class diagram

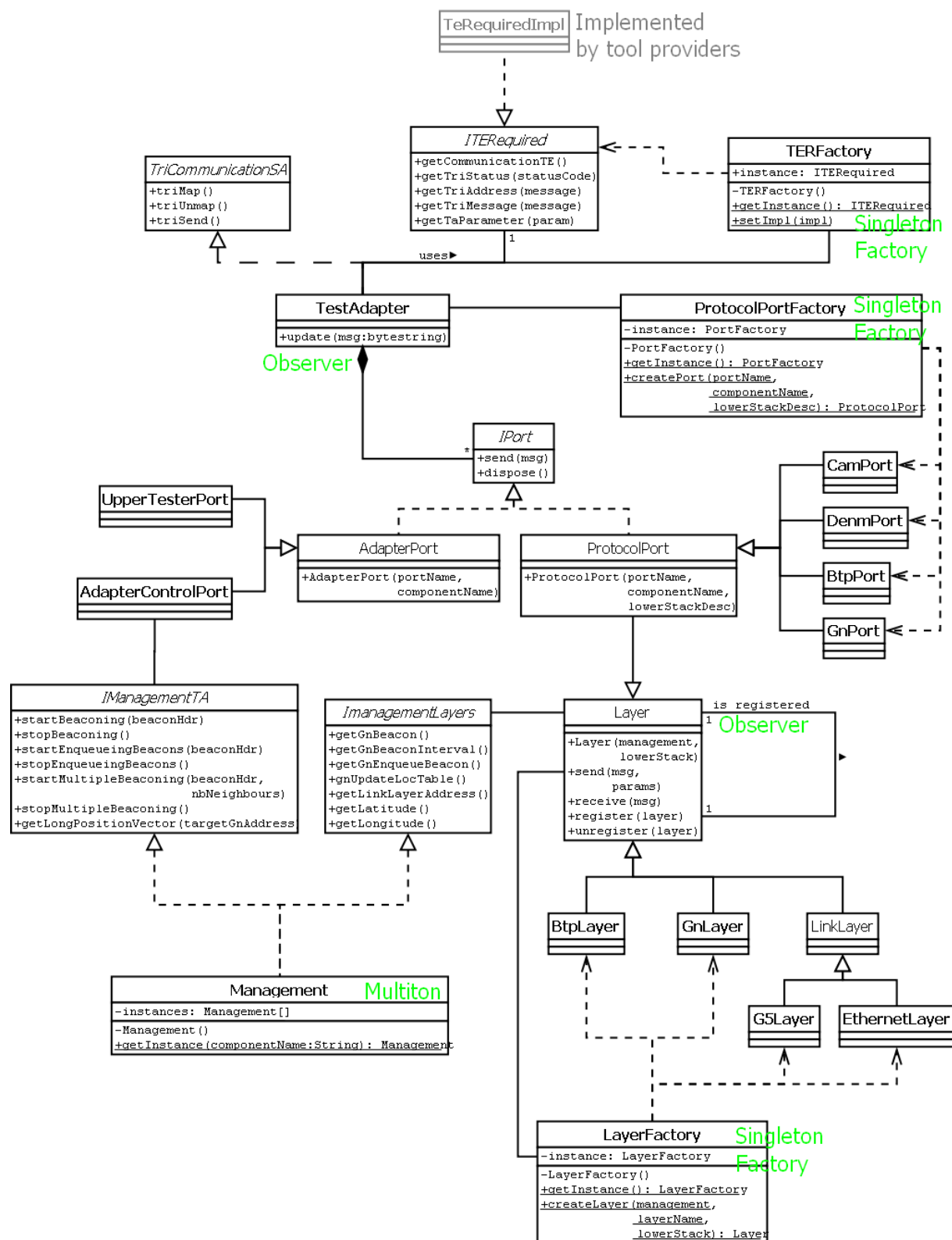


Figure E.1: Test adapter complete class diagram

Annex F:

Bibliography

This annex lists all test specifications which were integrated with the Conformance Validation Framework:

- ETSI TS 102 868-1 (V1.2.1): "Intelligent Transport Systems (ITS); Testing; Conformance test specification for Co-operative Awareness Messages (CAM); Part 1: Test requirements and Protocol Implementation Conformance Statement (PICS) proforma".
- ETSI TS 102 868-2 (V1.2.1): "Intelligent Transport Systems (ITS); Testing; Conformance test specification for Co-operative Awareness Messages (CAM); Part 2: Test Suite Structure and Test Purposes (TSS&TP)".
- ETSI TS 102 869-1 (V1.2.1): "Intelligent Transport Systems (ITS); Testing; Conformance test specification for Of Decentralized Environmental Notification basic Service (DENM); Part 1: Test requirements and Protocol Implementation Conformance Statement (PICS) proforma".
- ETSI TS 102 869-2 (V1.3.1): "Intelligent Transport Systems (ITS); Testing; Conformance test specification for Of Decentralized Environmental Notification basic Service (DENM); Part 2: Test Suite Structure and Test Purposes (TSS&TP)".
- ETSI TS 102 870-1 (V1.3.1): "Intelligent Transport Systems (ITS); Testing; Conformance test specifications for Geonetworking Basic Transport Protocol (BTP); Part 1: Test requirements and Protocol Implementation Conformance Statement (PICS) proforma".
- ETSI TS 102 870-2 (V1.1.1): "Intelligent Transport Systems (ITS); Testing; Conformance test specifications for Geonetworking Basic Transport Protocol (BTP); Part 2: Test Suite Structure and Test Purposes (TSS&TP)".
- ETSI TS 102 859-1 (V1.2.1): "Intelligent Transport Systems (ITS); Testing; Conformance test specifications for Transmission of IP packets over GeoNetworking; Part 1: Test requirements and Protocol Implementation Conformance Statement (PICS) proforma".
- ETSI TS 102 859-2 (V1.2.1): "Intelligent Transport Systems (ITS); Testing; Conformance test specifications for Transmission of IP packets over GeoNetworking; Part 2: Test Suite Structure and Test Purposes (TSS&TP)".
- ETSI EG 202 798 (V1.1.1): "Intelligent Transport Systems (ITS); Testing; Framework for conformance and interoperability testing".
- JNetPcap library: "<http://jnetpcap.com/>".
- ETSI ES 201 873-1 (V4.5.1): "Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 1: TTCN-3 Core Language".
- ETSI ES 201 873-6 (V4.5.1): "Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 6: TTCN-3 Control Interface (TCI)".

List of figures

Figure 1: General architecture	9
Figure 2: Communication via G5 adaptation box	10
Figure 3: Communication via Ethernet	10
Figure 4: Codec class diagram	12
Figure 5: Message sending sequence diagram	14
Figure 6: Message reception sequence diagram	14
Figure 7: Test Adapter class diagram.....	15
Figure 8: Port initialization sequence diagram	16
Figure 9: Upper Tester architecture.....	18
Figure E.1: Test adapter complete class diagram	47

History

Document history		
V1.1.1	November 2012	Publication
V1.2.1	May 2014	Publication