ETSI TR 103 099 V1.1.1 (2012-11)

Technical Report

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

Reference

DTR/ITS-0030038

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

Copyright Notification

Contents

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 System (ITS).

Introduction

In response to EC mandate M/453, 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, 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 CAM, DENM, BTP and GeoNetworking abstract test suites (ATS) ([i.10] to [i.13]). The results of the validation are described in the TR 103 061 series [i.4] to [i.8].

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 (V4.2.1): "Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 5: TTCN-3 Runtime Interface (TRI)".

[i.2] ETSI ES 201 873-6 (V4.2.1): "Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 6: TTCN-3 Control Interface (TCI)".

[i.3] ETSI EG 201 015 (V2.1.1): "Methods for Testing and Specification (MTS); Standards engineering process; A handbook of validation methods".

[i.4] ETSI TR 103 061-1 (V1.1.1): "Intelligent Transport Systems (ITS); Testing; Conformance test specification for Co-operative Awareness Messages (CAM); CAM validation report".

[i.5] ETSI TR 103 061-2 (V1.1.1): "Intelligent Transport Systems (ITS); Testing; Conformance test specification for Decentralized Environmental Notification basic Service Message (DENM); DENM validation report".

[i.6] ETSI TR 103 061-3 (V1.1.1): "Intelligent Transport Systems (ITS); Testing; Conformance test specification for Geographical addressing and forwarding for point-to-point and point-to-multipoint communications; GeoNetworking validation report".

[i.7] ETSI TR 103 061-4 (V1.1.1): "Intelligent Transport Systems (ITS); Testing; Conformance test specification for GeoNetworking Basic Transport Protocol (BTP); GeoNetworking BTP validation report".

[i.8] ETSI TR 103 061-5 (V1.1.1): "Intelligent Transport Systems (ITS); Testing; IPv6 over GeoNetworking validation report".

[i.9] 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.10] ETSI TS 102 859-3 (V.1.1.1): "Intelligent Transport Systems (ITS); Testing; Conformance test specification for Transmission of IP packets over GeoNetworking; Part 3: Abstract Test Suite (ATS) and Protocol Implementation eXtra Information for Testing (PIXIT)".

[i.11] ETSI TS 102 868-3 (V.1.1.1): "Intelligent Transport Systems (ITS); Testing; Conformance test specification for Co-operative Awareness Messages (CAM); Part 3: Abstract Test Suite (ATS) and Protocol Implementation eXtra Information for Testing (PIXIT)".

[i.12] ETSI TS 102 869-3 (V.1.1.1): "Intelligent Transport Systems (ITS); Testing; Conformance test specification for Of Decentralized Environmental Notification basic Service (DENM); Part 3: Abstract Test Suite (ATS) and Protocol Implementation eXtra Information for Testing (PIXIT)".

[i.13] 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)".

3 Abbreviations

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

AC	Adapter Control
API	Application Programming Interface
ASN	Abstract Syntax Notation
ATS	Abstract Test Suite
BTP	Basic Transport Protocol
CAM	Cooperative Awareness Message
CT	Causer Type
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
LB	Low Beam
LDM	Local Dynamic Map
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
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.3]);
- 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.

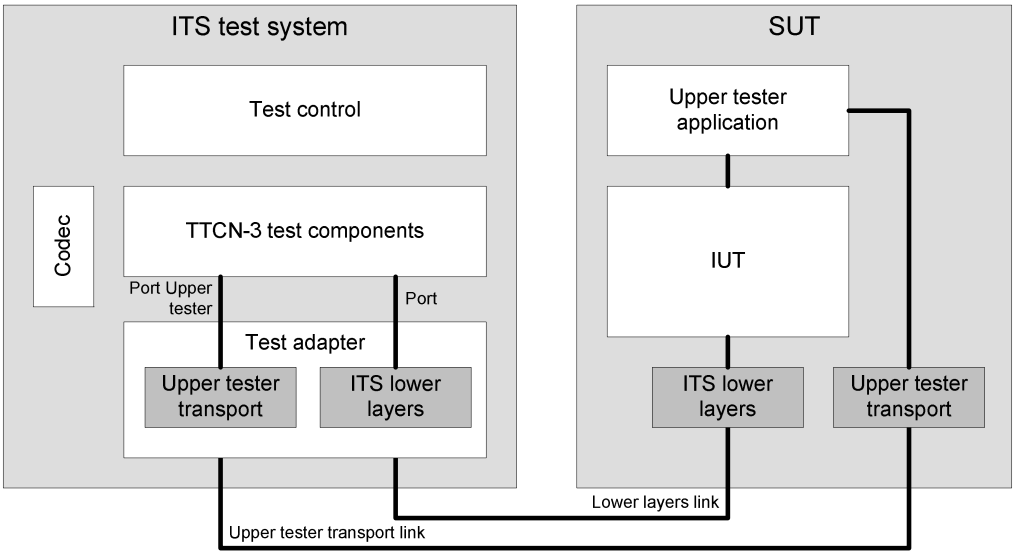


Figure 1: General architecture

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

The ITS test platform is composed of two hardware equipments, a standard PC and a G5 adapter box.

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 synchronisation between the SUT and the test system.
- Test system processes (especially the test adapter) have to be granted unrestricted control to telecommunication hardware.

Time synchronisation 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.

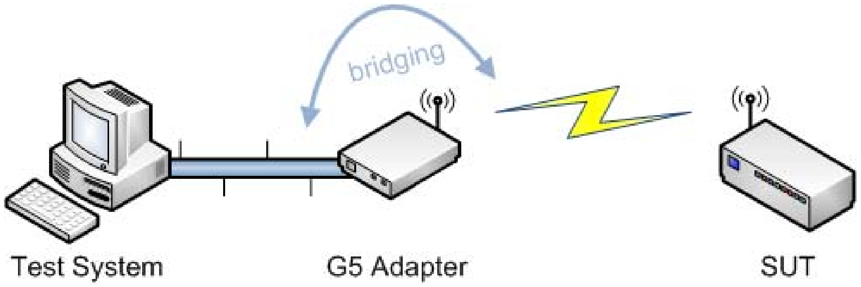


Figure 2: Communication via G5 adaptation box

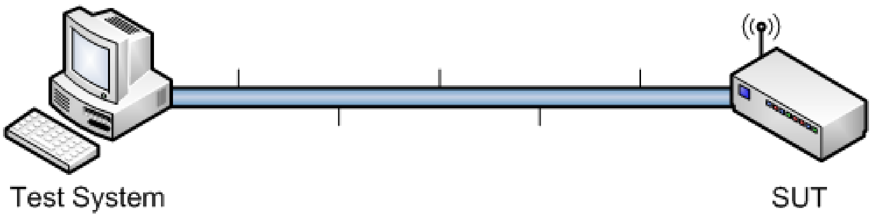


Figure 3: Communication via Ethernet

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 Wireless^{TM} MK2 has been chosen to fulfil this task. This device is fully IEEE 802.11p [i.9] 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. The source code of this program and its configuration file are provided in annex D.

6 Codecs

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

- One independent codec per ITS protocol.
- One codec 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 (GeoNetworkingInd, GeoNetworkingReq, etc.).

For protocol messages defined in ASN.1 (CAM and DENM protocols), usage of dedicated commercial ASN.1 tools is recommended.

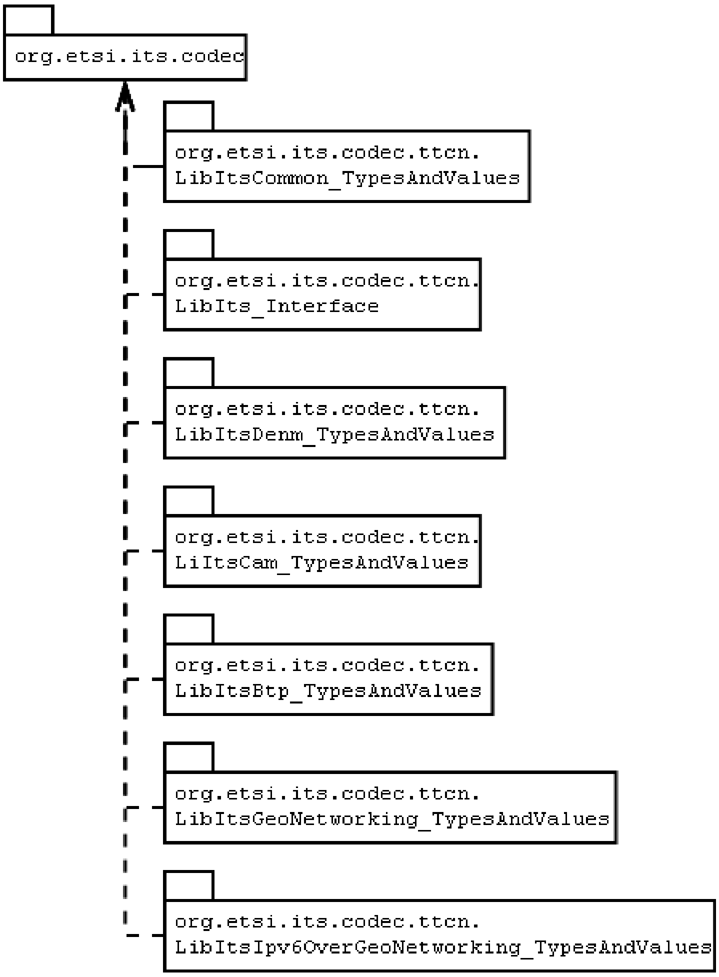


Figure 4: Codec package organisation

Selection of correct codec for encoding a message at runtime is dictated by means of the "with encode" statement within TTCN-3 modules. For instance the following statement:

```
with {
 encode "LibItsGeoNetworking_TypesAndValues"
  }
```

will cause `org.etsi.its.codec.ttcn.LibItsGeoNetworking_TypesAndValuesCodec` to be invoked.

6.1 Advanced details of implementation

Figure 5 gives an overview of the relations between the different java classes implementing the codec. The structure is relatively simple. Connection with the tool-dependent classes is realized through the ICodec interface and is not depicted in figure 5.

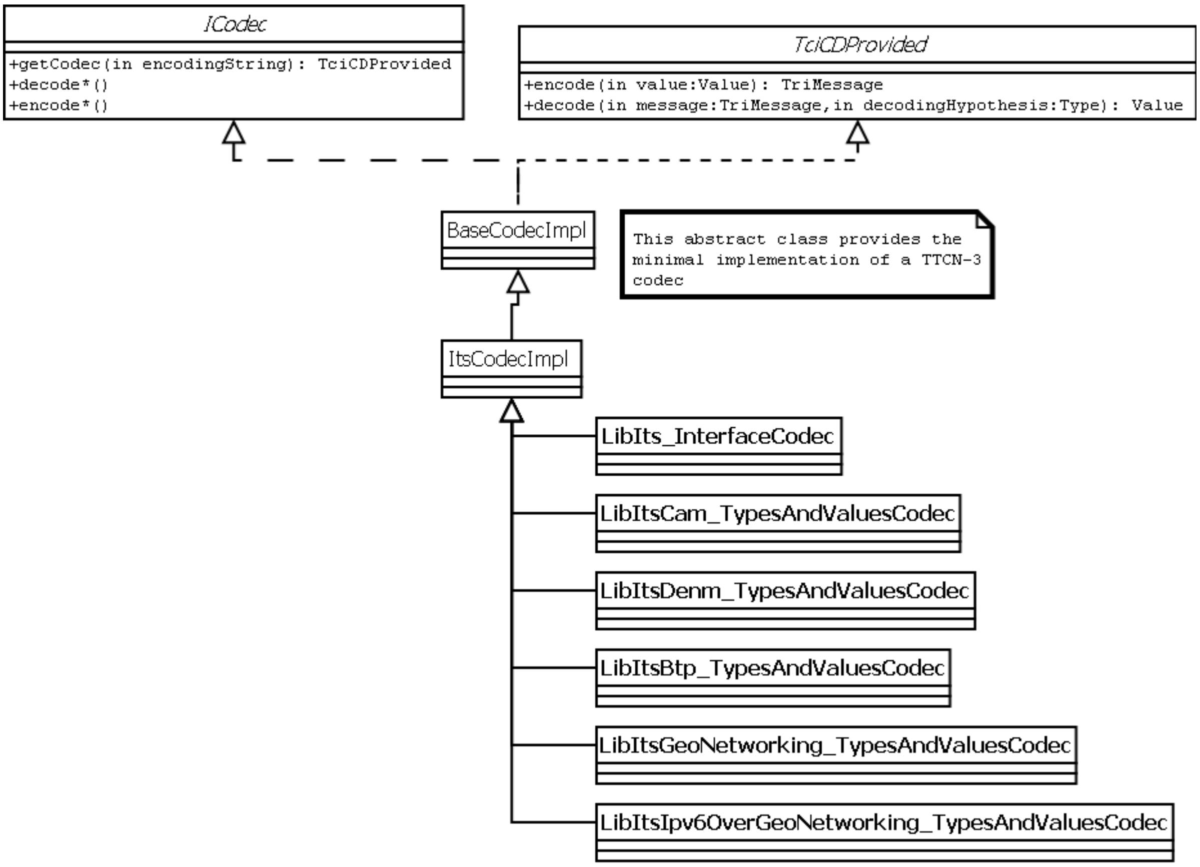

ICodec
+getCodec(in encodingString): TciCDProvided
+decode*()
+encode*()
TciCDProvided
+encode(in value:Value): TriMessage
+decode(in message:TriMessage,in decodingHypothesis:Type): Value
BaseCodecImpl
This abstract class provides the minimal implementation of a TTCN-3 codec
ItsCodecImpl
LibIts_InterfaceCodec
LibItsCam_TypesAndValuesCodec
LibItsDenm_TypesAndValuesCodec
LibItsBtp_TypesAndValuesCodec
LibItsGeoNetworking_TypesAndValuesCodec
LibItsIpv6OverGeoNetworking_TypesAndValuesCodec

Figure 5: Codec class diagram

Each codec implements the standard TCI interface TciCDProvided as described in ES 201 873-6 [i.2]. In addition codecs have to implement the ICodec interface, which is providing a tool-independent instantiation API to TTCN-3 tools.

The BaseCodecImpl class implements the minimal functionalities of a codec and is used as a base class for further codec development. For extensibility purpose, this class is not ITS-specific, and it can be used as-is in other platform projects.

The ItsCodecImpl class directly extends BaseCodecImpl and provides ITS common codec functionalities. Each ITS codec derives from this class.

7 Test Adapter

The test adapter conceptually splits into three parts:

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

7.1 Lower Tester

7.1.1 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 standardised 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 E). 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 the next section, 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 6 and 7 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.

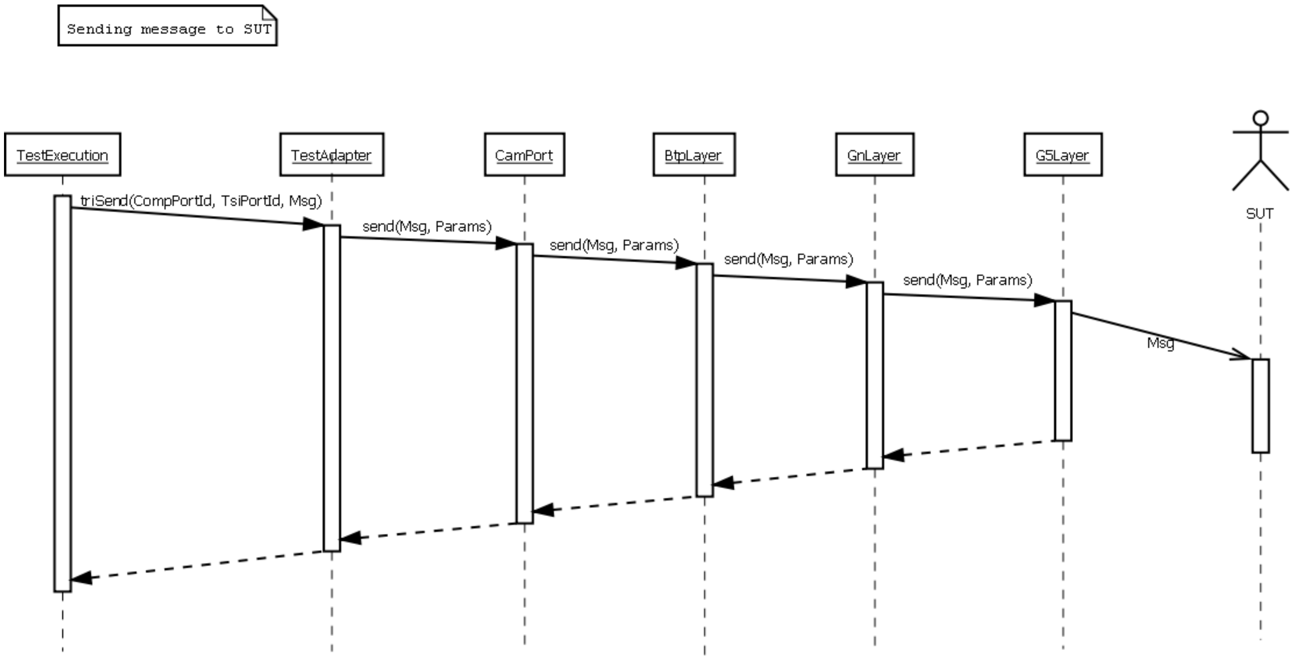


Figure 6: Message sending sequence diagram

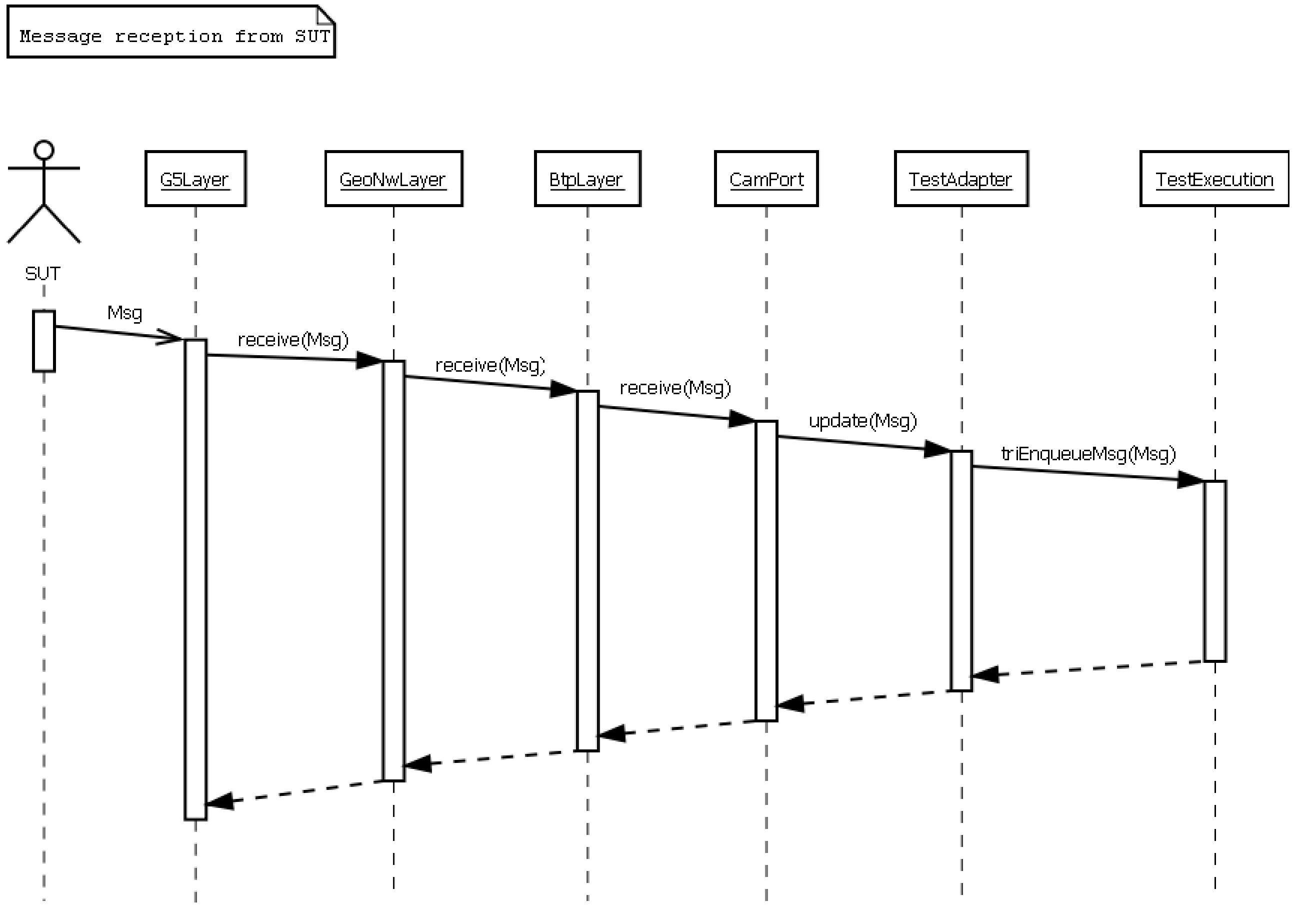


Figure 7: Message reception sequence diagram

7.1.2 Advanced details of implementation

Figure 8 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 F.

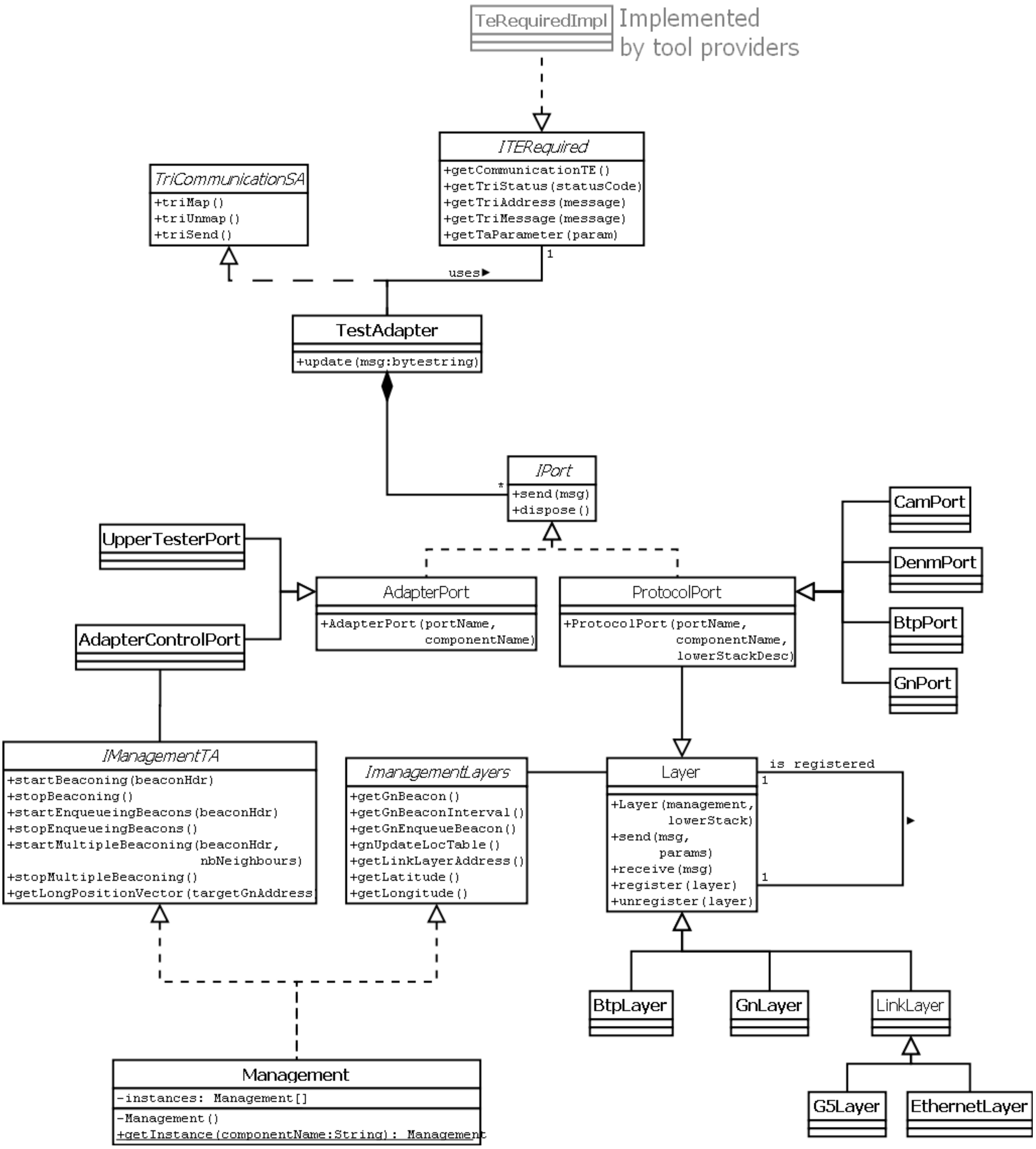


Figure 8: Test Adapter class diagram

The main class of this implementation is the `TestAdapter` class. It implements the standardised 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 realising communication between TTCN-3 and SUT.
- Adapter ports which are used for test adapter configuration and upper tester communication.

Upon protocol port initialisation, lower layers are instanciated in cascade and chained as depicted in figure 9, base on lower stack description. This picture also illustrate the usage of "Factory" design patterns for instanciating `Layer` and `Port` objects. Each `Layer` is responsible for encapsulating/decapsulating packets and transmitting result to lower/upper layer using `send()`/`receive()` methods.

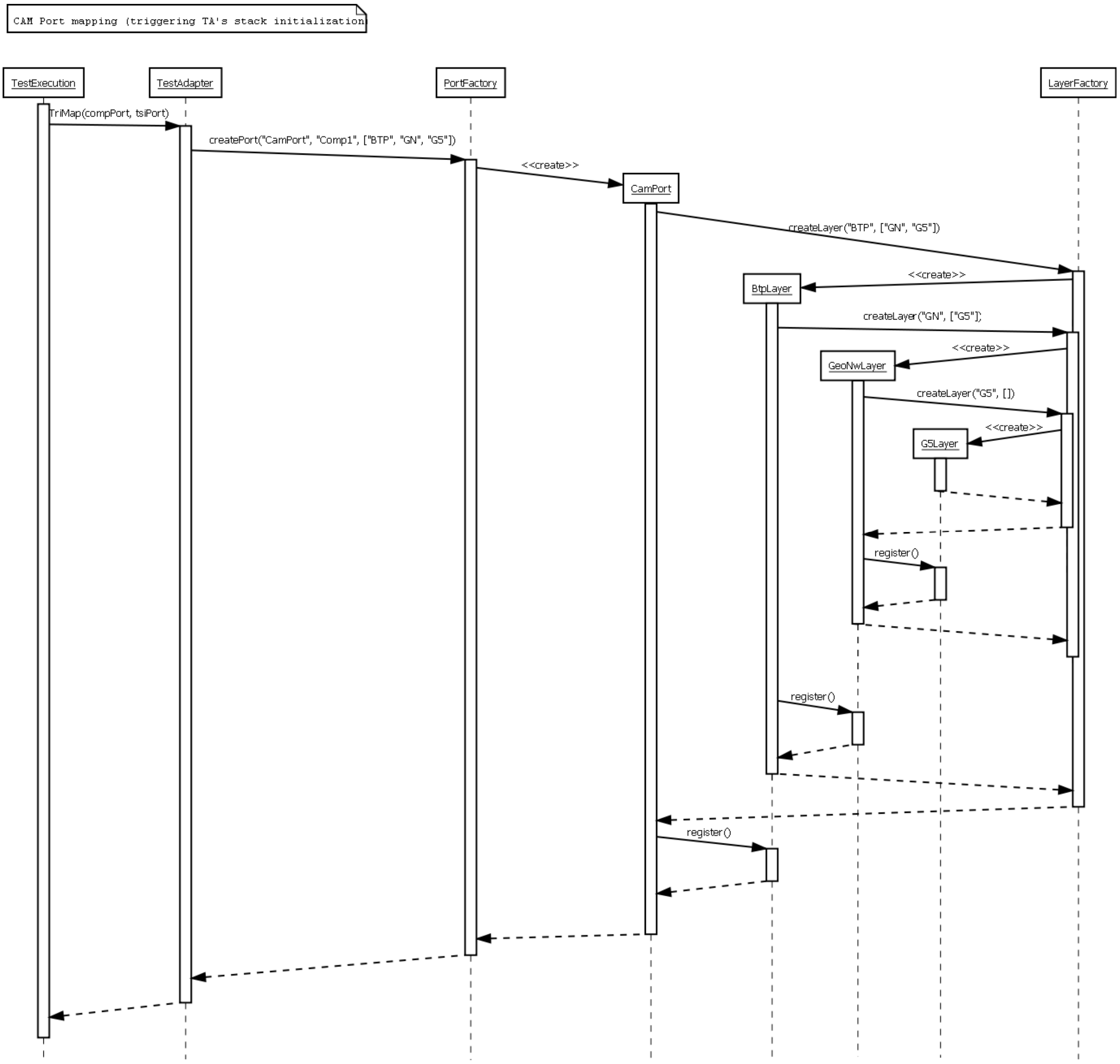

CAM Port mapping (triggering TA's stack initialization
TestExecution
TestAdapter
PortFactory
LayerFactory
TriMap(compPort, tsiPort)
createPort("CamPort", "Comp1", ["BTP", "GN", "G5"])
<<create>>
CamPort
createLayer("BTP", ["GN", "G5"])
<<create>>
BtpLayer
createLayer("GN", ["G5"];
<<create>>
GeoNwLayer
createLayer("G5", [])
<<create>>
G5Layer
register()
register()
register()

Figure 9: Port initialisation sequence diagram

Currently the following layers have been implemented:

- GnLayer: basic functionalities of GeoNetworking layer, including beaconing.
- BtpLayer.
- EthernetLayer. Important: 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 instanciated per TTCN-3 component.

7.1.3 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.

7.1.4 Adapter Control primitives

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

AC 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

7.2 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.

7.3 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. 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 module 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.

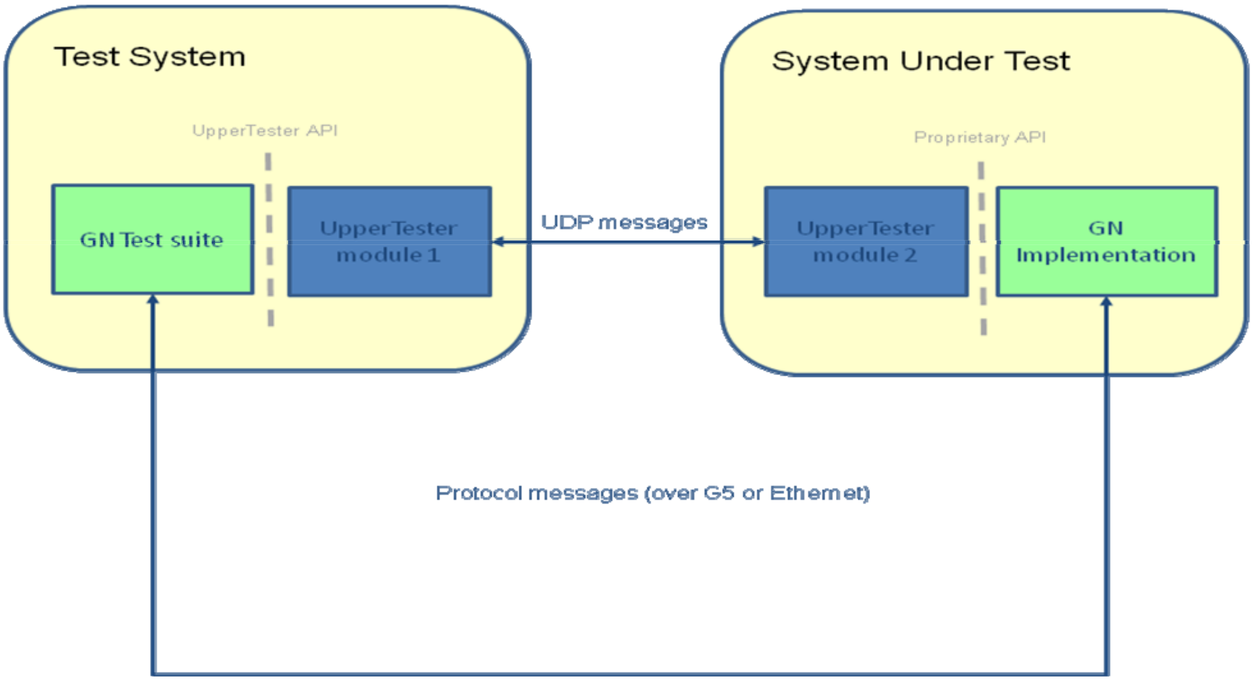


Figure 10: Upper Tester architecture

The ITS test platform is provided with three different test system side implementations. The different implementations can be selected by setting item 7 of table E.1 to the following values:

- `Yes`: This upper tester module does not require any counterpart running on SUT. It will report all upper tester calls to be successful without performing any action. This upper tester module is only useful for debugging purposes.
- `Operator`: This upper tester module does not require any counterpart running on SUT. For each upper tester call it will display action information in a dialog box so that the test operator can manipulate the SUT.
- `Generic`: This upper tester module implements the upper tester message based API described in annex C.

The source code of the test system side of the upper tester modules is provided in annex B as part of the test adapter source code.

Annex A: Codecs source code

The software modules are contained in archive AnnexA_Codec.zip which is included in archive tr_103099v010101p0.zip which accompanies the present document.

Annex B:
Test Adapter source code

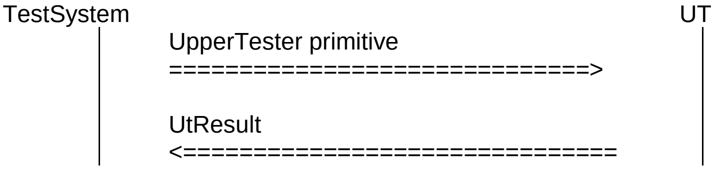
The software modules are contained in archive AnnexB_Adapter.zip which is included in archive tr_103099v010101p0.zip which accompanies the present document.

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).

The message exchange always occurs as described below:



Message exchange is always initiated by TestSystem. Upper Tester always replies to UpperTester primitives by sending an UpperTesterResult message indicating Success or Failure of the 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 CAM Upper Tester Primitives

C.1.1 CamInitialize

This message is used to request initialization of CAM implementation.

0
0 1 2 3 4 5 6 7
MessageType = 0x2E

Name	Length	Value
MessageType	1 byte	0x2E

C.1.2 ChangeHeading

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

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 = 0x70								Direction															

Name	Length	Value
MessageType	1 byte	0x70
Direction	2 bytes	Integer value from 0 to 28 799 0: North 7 200: East 14 400: South 21 600: West

C.1.3 ChangePosition

This message is used to change the position of the ITS station. The new position of the ITS station is at 'Distance' meters of previous position.

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								Distance																							

Name	Length	Value
MessageType	1 byte	0x71
Distance	4 bytes	Position offset in meters

C.1.4 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 × 300 = 13 m/s.

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 = 0x72								SpeedVariation																							

Name	Length	Value
MessageType	1 byte	0x72
SpeedVariation	4 bytes	Speed variation in units of 0,01 m/s

C.1.5 SetCrashSignal

This message is used to set the crash signal of the ITS station.

0								1							
0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
MessageType = 0x73								CrashStatus							

Name	Length	Value
MessageType	1 byte	0x73
CrashStatus	1 byte	0x00: False 0x01: True Other values are forbidden

C.1.6 SetDangerousGoodsStatus

This message is used to set the dangerous goods status of the ITS station.

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 = 0x74								DangerousGoodsStatus															

Name	Length	Value
MessageType	1 byte	0x74
DangerousGoodsStatus	2 bytes	Integer value from 0 to 8 191 0: no dangerous goods

C.1.7 SetLengthWidthPrecision

This message is used to set the length and width precision of the ITS station.

0								1							
0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
MessageType = 0x75								Precision							

Name	Length	Value
MessageType	1 byte	0x75
Precision	1 byte	0x00: Length/Width precision is not precisely determined 0x01: Length/Width precision is precisely determined Other values are forbidden

C.1.8 SetDistanceToStopLine

This message is used to set the distance to next stop line of the ITS station.

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 = 0x76								Distance															

Name	Length	Value
MessageType	1 byte	0x76
Distance	2 bytes	Distance in units of 1 meter From 0 to 65 535

C.1.9 SetTurnAdvice

This message is used to set the turn advice of the ITS station.

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 = 0x77								Direction								Distance															

Name	Length	Value
MessageType	1 byte	0x77
Direction	1 byte	0: uTurn 1: sharpRight 2: right 3: slightRight 4: straight 5: slightLeft 6: left 7: sharpLeft
Distance	2 bytes	Distance in units of 1 meter From 0 to 65 535

C.1.10 SetCurvature

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

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 = 0x78								Curvature															

Name	Length	Value
MessageType	1 byte	0x78
Curvature	2 bytes	Signed integer. Curvature change from -1 023 to 1 023

C.1.11 SetOccupancy

This message is used to set the ITS station occupancy.

0								1							
0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
MessageType = 0x79								Occupancy							

Name	Length	Value
MessageType	1 byte	0x79
Occupancy	1 byte	Occupancy value from 0 to 255

C.1.12 SetDoorStatus

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

0								1							
0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
MessageType = 0x7A								Reserved				D	P	M	L

Name	Length	Value
MessageType	1 byte	0x7A
Reserved	4 bits	Set to 0
D	1 bit	0: driver door closed 1: driver door opened
P	1 bit	0: passenger door closed 1: passenger door opened
M	1 bit	0: maintenance door closed 1: maintenance door opened
L	1 bit	0: luggage door closed 1: luggage door opened

C.1.13 SetLightBarStatus

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

0								1							
0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
MessageType = 0x7B								Status							

Name	Length	Value
MessageType	1 byte	0x7B
Status	1 byte	0: unavailable 1: disabled 2: enabled 3: engaged

C.1.14 SetSireneStatus

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

0								1							
0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
MessageType = 0x7C								Status							

Name	Length	Value
MessageType	1 byte	0x7C
Status	1 byte	0: unavailable 1: disabled 2: enabled 3: engaged

C.1.15 SetTrafficLightPriority

This message is used to set the traffic light priority of the ITS station.

0	1
0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7
MessageType = 0x7D	TrafficLightPriority

Name	Length	Value
MessageType	1 byte	0x7D
TrafficLightPriority	1 byte	Traffic light priority from 0 to 7

C.1.16 SetScheduleDeviation

This message is used to set the schedule deviation of the ITS station.

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 = 0x7E	Delay	

Name	Length	Value
MessageType	1 byte	0x7E
Delay	2 bytes	Signed integer. Schedule deviation from -900 to 3 600

C.1.17 SetPtLineDescription

This message is used to set the public transport line description of the ITS station.

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 = 0x82	CourseLength	CourseOfJourney	
	...		
LineLength	LineRef		
	...		
RouteLength	RouteRef		
	...		

Name	Length	Value
MessageType	1 byte	0x82
CourseLength	1 byte	Length of 'CourseOfJourney' field
CourseOfJourney	Variable	String. Course of Journey
LineLength	1 byte	Length of 'LineRef' field
LineRef	Variable	String. Line Reference
RouteLength	1 byte	Length of 'RouteRef' field
RouteRef	Variable	String. Route Reference

C.1.18 SetExteriorLightsStatus

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 = 0x7F	LB	HB	LT	RT	A	D	F	P

Name	Length	Value
MessageType	1 byte	0x7F
LB	1 bit	0: low beam headlights off 1: low beam headlights on
HB	1 bit	0: high beam voff 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
A	1 bit	0: automatic light control off 1: automatic light control on
D	1 bit	0: daytime running lights off 1: daytime running lights on
F	1 bit	0: fog light off 1: fog light on
P	1 bit	0: parking lights off 1: parking lights on

C.1.19 CheckCam

This message is used to check whether information contained in a CAM PDU has been transmitted to upper layer (LDM).

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 = 0x86	CamPduLength		CamPdu
	...		

Name	Length	Value
MessageType	1 byte	0x86
CamPduLength	2 bytes	Length of 'CamPdu' field
CamPdu	Variable	CamPdu to be checked

C.2 DENM Upper Tester Primitives

C.2.1 DenmInitialize

This message is used to request initialization of DENM implementation.

0
0 1 2 3 4 5 6 7
MessageType = 0x39

Name	Length	Value
MessageType	1 byte	0x39

C.2.2 GenerateDenmEvent

This message is used to create a new DENM event.

0 0 1 2 3 4 5 6 7	1 0 1 2 3 4 5 6 7	2 0 1 2 3 4 5 6 7	3 0 1 2 3 4 5 6 7
MessageType = 0x3A	Cause	Subcause	TrafficFlowEffect
Frequency	Severity	ExpirationTime	
M CT Rel T	LinkedCause	LinkedSubCause	

Name	Length	Value
MessageType	1 byte	0x3A
Cause	1 byte	Event cause ID
Subcause	1 byte	Event sub-cause ID
TrafficFlowEffect	1 byte	Traffic flow effect
Frequency	1 byte	Repetition frequency in units of 0,1 Hz
Severity	1 byte	Severity of the event
ExpirationTime	6 bytes	Timestamp indicating event expiration in units of milliseconds
M	1 bit	0: the event is not mobile 1: the event is mobile
CT	3 bits	Cause type. 0: itsStation 1: geographical region Other values are forbidden
Rel	3 bits	Relevance. 0: physically relevant 1: difficult driving conditions Other values are forbidden
T	1 bit	Time critical. 0: not critical 1: time critical road safety event
LinkedCause	1 byte	Linked event cause
LinkedSubCause	1 byte	Linked event sub-cause

C.2.3 CancelDenmEvent

This message is used to cancel an existing DENM event.

0 0 1 2 3 4 5 6 7	1 0 1 2 3 4 5 6 7	2 0 1 2 3 4 5 6 7	3 0 1 2 3 4 5 6 7
MessageType = 0x8d	Cause	Subcause	StationId
...			SequenceNo
...			

Name	Length	Value
MessageType	1 byte	0x8d
Cause	1 byte	Event cause ID
Subcause	1 byte	Event sub-cause ID
StationId	4 bytes	Original Station ID
SequenceNo	2 bytes	Event sequence number

C.2.4 NegateDenmEvent

This message is used to negate an existing DENM event.

0 0 1 2 3 4 5 6 7	1 0 1 2 3 4 5 6 7	2 0 1 2 3 4 5 6 7	3 0 1 2 3 4 5 6 7
MessageType = 0x8a	Cause	Subcause	StationId
...			SequenceNo
...			

Name	Length	Value
MessageType	1 byte	0x8a
Cause	1 byte	Event cause ID
Subcause	1 byte	Event sub-cause ID
StationId	4 bytes	Original Station ID
SequenceNo	2 bytes	Event sequence number

C.2.5 UpdateDenmEvent

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

0 0 1 2 3 4 5 6 7	1 0 1 2 3 4 5 6 7	2 0 1 2 3 4 5 6 7	3 0 1 2 3 4 5 6 7
MessageType = 0x8f	Cause	Subcause	StationId
...			SequenceNo
...			ExpirationTime
...			
...			

Name	Length	Value
MessageType	1 byte	0x8f
Cause	1 byte	Event cause ID
Subcause	1 byte	Event sub-cause ID
StationId	4 bytes	Original Station ID
SequenceNo	2 bytes	Event sequence number
ExpirationTime	6 bytes	Timestamp indicating event expiration in units of milliseconds

C.2.6 CheckDenm

This message is used to check whether information contained in a DENM PDU has been transmitted to upper layer (LDM).

0 0 1 2 3 4 5 6 7	1 0 1 2 3 4 5 6 7	2 0 1 2 3 4 5 6 7	3 0 1 2 3 4 5 6 7
MessageType = 0x46	DenmPduLength		DenmPdu
	...		

Name	Length	Value
MessageType	1 byte	0x46
DenmPduLength	2 bytes	Length of 'DenmPdu' field
DenmPdu	Variable	DenmPdu to be checked

C.3 GeoNetworking Upper Tester Primitives

C.3.1 GnInitialize

This message is used to request initialization of GeoNetworking implementation.

0							
0	1	2	3	4	5	6	7
MessageType = 0x25							

Name	Length	Value
MessageType	1 byte	0x25

C.3.2 GenerateGeoUnicast

This message is used to trigger a GeoUnicast message.

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 = 0x28								DstGnAddress																							
...																															
...								Lifetime																PayloadLength							
...								Payload																							

Name	Length	Value
MessageType	1 byte	0x28
DstGnAddr	8 bytes	Destination GN Address
Lifetime	2 bytes	Packet lifetime in miliseconds
PayloadLength	2 bytes	Length of 'Payload' field
Payload	Variable	Packet's final payload

C.3.3 GenerateGeoAnycast

This message is used to trigger a GeoAnycast message.

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 = 0x2a								Shape								Lifetime															
Latitude																															
Longitude																															
DistanceA																DistanceB															
Angle																PayloadLength															
Payload																															

Name	Length	Value
MessageType	1 byte	0x2a
Shape	1 byte	0: Circle 1: Rectangle 2: Ellipse
Lifetime	2 bytes	Packet lifetime in miliseconds
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

C.3.4 GenerateGeoBroadcast

This message is used to trigger a GeoBroadcast message.

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 = 0x29	Shape	Lifetime	
Latitude			
Longitude			
DistanceA		DistanceB	
Angle		PayloadLength	
Payload			

Name	Length	Value
MessageType	1 byte	0x29
Shape	1 byte	0: Circle 1: Rectangle 2: Ellipse
Lifetime	2 bytes	Packet lifetime in miliseconds
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

C.3.5 GenerateSHB

This message is used to trigger a SHB message.

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 = 0x2b	PayloadLength		Payload

Name	Length	Value
MessageType	1 byte	0x2b
PayloadLength	2 bytes	Length of 'Payload' field
Payload	Variable	Packet's final payload

C.3.6 GenerateTSB

This message is used to trigger a TSB message.

<table>
<tr><td colspan="8">0</td><td colspan="8">1</td><td colspan="8">2</td><td colspan="8">3</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr>
<tr><td colspan="8">MessageType = 0x2c</td><td colspan="8">NbHops</td><td colspan="16">PayloadLength</td></tr>
<tr><td colspan="32">Payload</td></tr>
</table>

Name	Length	Value
MessageType	1 byte	0x2c
NbHops	1 byte	Number of hops
PayloadLength	2 bytes	Length of 'Payload' field
Payload	Variable	Packet's final payload

C.3.7 CheckPacket

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

<table>
<tr><td colspan="8">0</td><td colspan="8">1</td><td colspan="8">2</td><td colspan="8">3</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr>
<tr><td colspan="8">MessageType = 0xFF</td><td colspan="16">PacketLength</td><td colspan="8">Packet</td></tr>
</table>

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

Annex D: MK2 bridge application

The software modules are contained in archive AnnexD_MK2_Bridge_Application.zip which is included in archive tr_103099v010101p0.zip which accompanies the present document.

Annex E: 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. Theorically the platform can be used above Linux based operating system, 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.

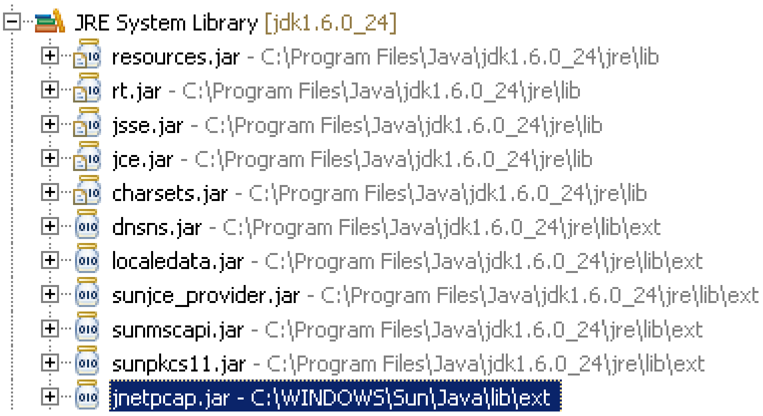
- Java™ JDK 1.6.0-24
  All the software used in the test platform have been developed using Java™ language.

- 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^{TM} MK2
  This device provides G5 connectivity to the test platform. This device features a G5 radion 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/received via the radio interface.

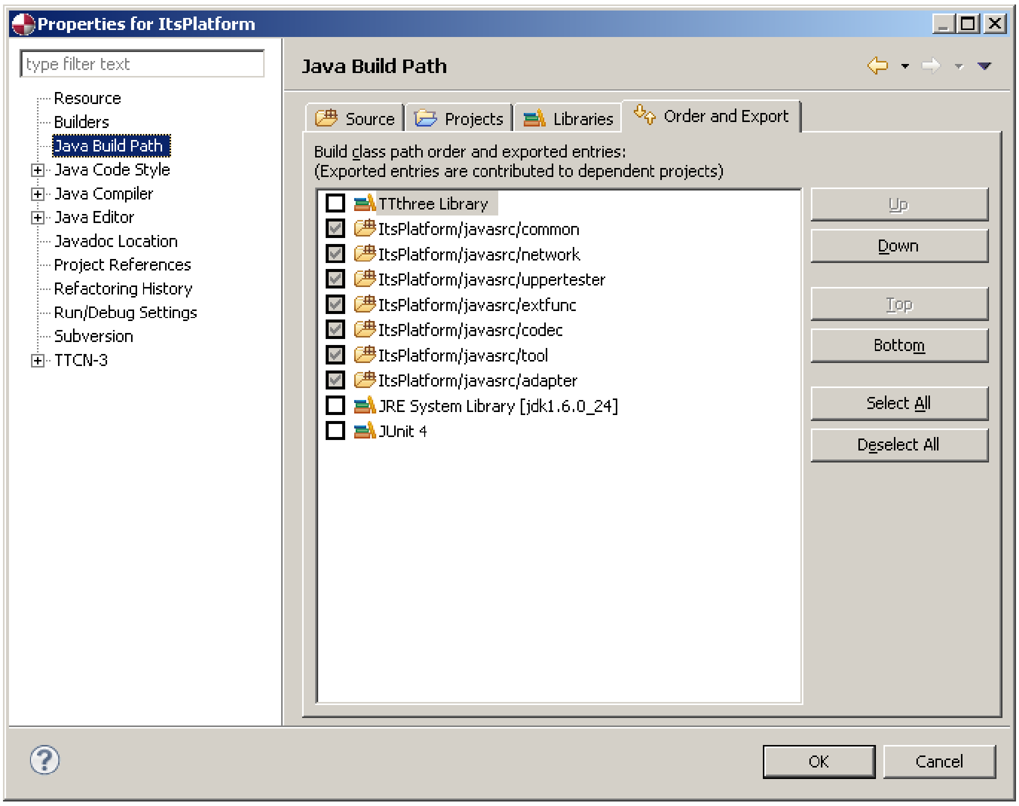
Before running successfully any test, a certain number of settings have to be verified in TTworkbench:

- 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:



```
JRE System Library [jdk1.6.0_24]
  resources.jar - C:\Program Files\Java\jdk1.6.0_24\jre\lib
  rt.jar - C:\Program Files\Java\jdk1.6.0_24\jre\lib
  jsse.jar - C:\Program Files\Java\jdk1.6.0_24\jre\lib
  jce.jar - C:\Program Files\Java\jdk1.6.0_24\jre\lib
  charsets.jar - C:\Program Files\Java\jdk1.6.0_24\jre\lib
  dnsns.jar - C:\Program Files\Java\jdk1.6.0_24\jre\lib\ext
  localedata.jar - C:\Program Files\Java\jdk1.6.0_24\jre\lib\ext
  sunjce_provider.jar - C:\Program Files\Java\jdk1.6.0_24\jre\lib\ext
  sunmscapi.jar - C:\Program Files\Java\jdk1.6.0_24\jre\lib\ext
  sunpkcs11.jar - C:\Program Files\Java\jdk1.6.0_24\jre\lib\ext
  jnetpcap.jar - C:\WINDOWS\Sun\Java\lib\ext
```

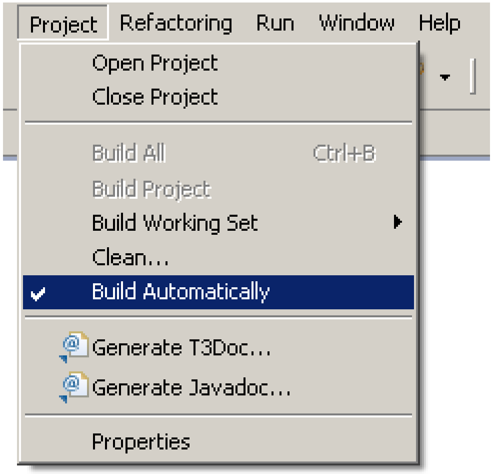
- Test Adapter and Codecs source folders have to be declared in project's Java Build Path:



Build class path order and exported entries:
(Exported entries are contributed to dependent projects)

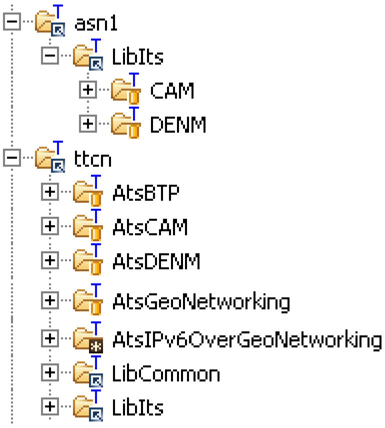
☐ TTthree Library
☑ ItsPlatform/javasrc/common
☑ ItsPlatform/javasrc/network
☑ ItsPlatform/javasrc/uppertester
☑ ItsPlatform/javasrc/extfunc
☑ ItsPlatform/javasrc/codec
☑ ItsPlatform/javasrc/tool
☑ ItsPlatform/javasrc/adapter
☐ JRE System Library [jdk1.6.0_24]
☐ JUnit 4

- 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

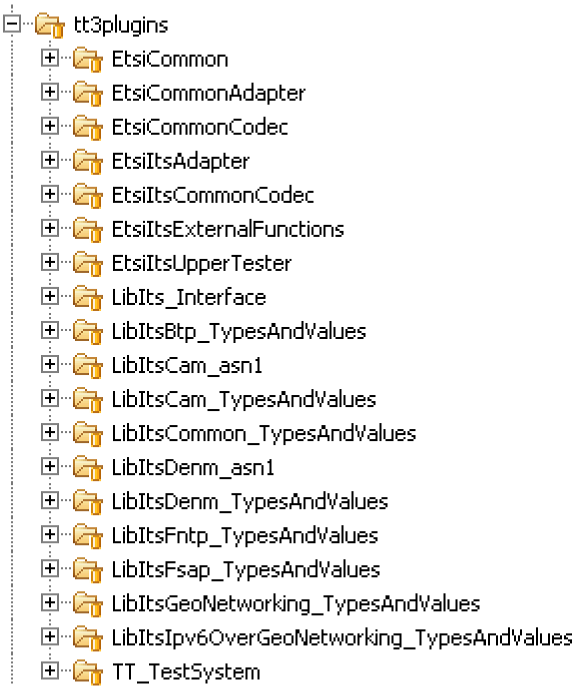
- TTCN-3 test suites and ASN.1 definitions have to be copied to the project and declared as TTCN-3 source folders:



```
asn1
    LibIts
        CAM
        DENM
ttcn
    AtsBTP
    AtsCAM
    AtsDENM
    AtsGeoNetworking
    AtsIPv6OverGeoNetworking
    LibCommon
    LibIts
```

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

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



```
tt3plugins
    EtsiCommon
    EtsiCommonAdapter
    EtsiCommonCodec
    EtsiItsAdapter
    EtsiItsCommonCodec
    EtsiItsExternalFunctions
    EtsiItsUpperTester
    LibIts_Interface
    LibItsBtp_TypesAndValues
    LibItsCam_asn1
    LibItsCam_TypesAndValues
    LibItsCommon_TypesAndValues
    LibItsDenm_asn1
    LibItsDenm_TypesAndValues
    LibItsFntp_TypesAndValues
    LibItsFsap_TypesAndValues
    LibItsGeoNetworking_TypesAndValues
    LibItsIpv6OverGeoNetworking_TypesAndValues
    TT_TestSystem
```

- 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="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 E.1 summarizes the authorised values for these parameters.

Table E.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 combinaison 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 65 635. 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 65 635.

Annex F:
Complete Test Adapter class diagram

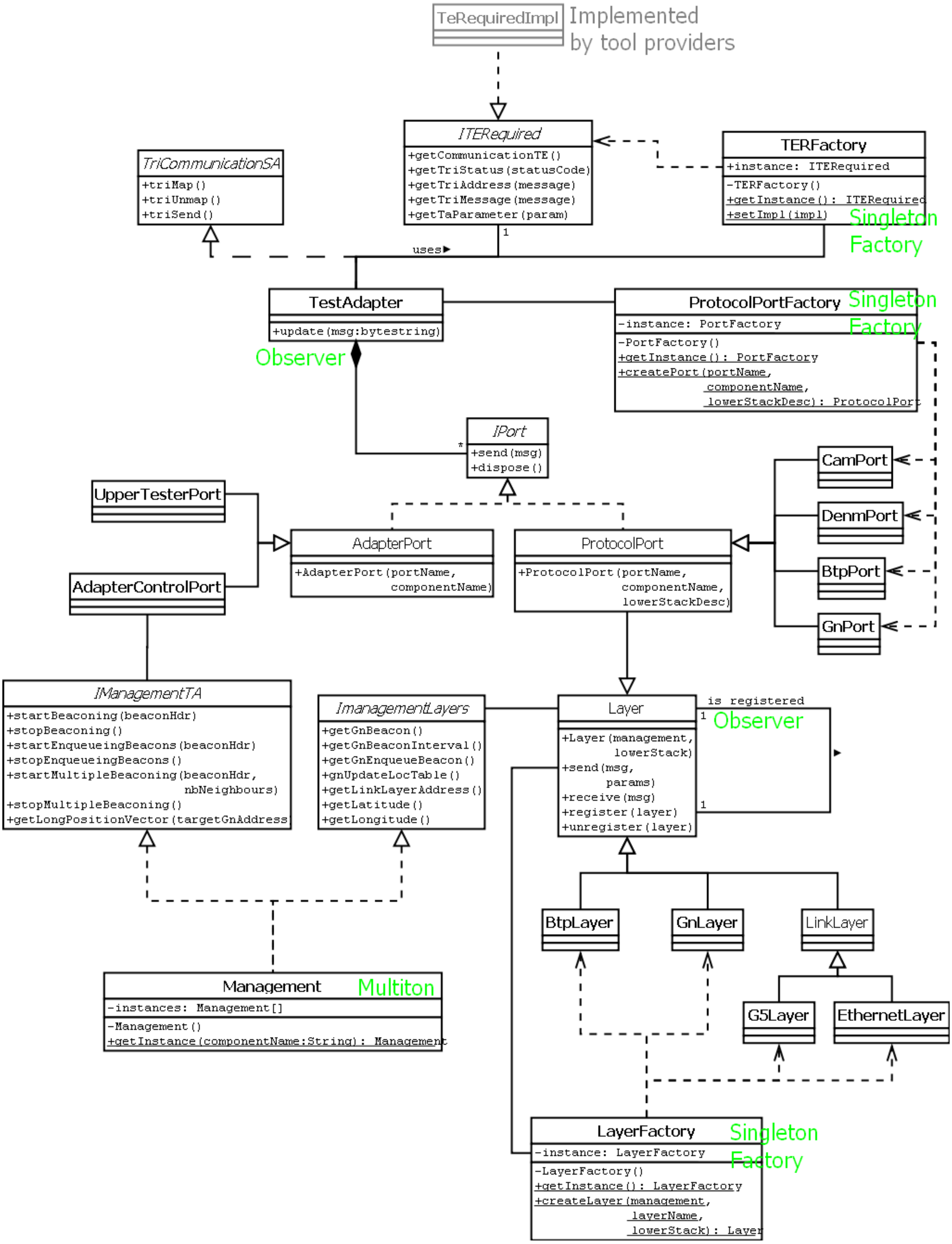


Figure F.1: Test adapter complete class diagram

Annex G: Validated ATS in TTCN-3 and ASN.1 modules

During the validation a number of issues were fixed in the ATSs of the different modules (refer to the appropriate reports TR 103 061-1 [i.4], -2 [i.5], -3 [i.6], -4 [i.7] and -5 [i.8]).

The software modules are contained in archive AnnexG_Validated_TTCN.zip which is included in archive tr_103099v010101p0.zip which accompanies the present document.

Annex H: Bibliography

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

- ETSI TS 102 868-1 (V1.1.1): "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 (V1.1.1): "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 (V.1.1.1): "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 (V.1.1.1): "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 (V.1.1.1): "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 (V.1.1.1): "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 (V.1.1.1): "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 (V.1.1.1): "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 ES 201 873-1 (V4.2.1): "Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 1: TTCN-3 Core Language".
- ETSI EG 202 798 (V1.1.1): "Intelligent Transport Systems (ITS); Testing; Framework for conformance and interoperability testing".
- JNetPcap library: "http://jnetpcap.com/".

List of figures

History

Document history		
V1.1.1	November 2012	Publication