ETSI

World Class Standards

Solve the Challenge of Interoperability!

Interoperability Best Practices

EDITION 2

Supporting ICT Standardization

www.etsi.org

www.plugtests.org

Content

1. Market Drivers for Interoperability

In a world of converging yet diverse technologies, complex ICT systems must communicate and interwork on all levels – this is interoperability. One of the key motives for the development of ICT standards is to facilitate interoperability between products in a multi-vendor, multi-network and multi-service environment. Standards need to be designed and tested to ensure that products and services complying with them do, indeed, achieve interoperability.

Interoperability ensures that users have a much greater choice of products and that manufacturers benefit from the economies of scale that a wider market brings. Testing and interoperability are therefore crucial factors in the success of modern technologies, and it is market demand that has ensured that interoperability has maintained its prominent position in standardization.

THE CHALLENGE

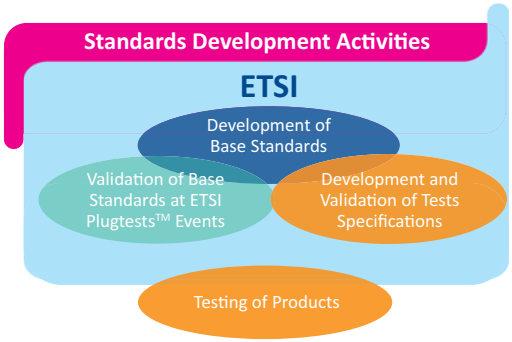
The trend towards a globally interconnected world, demonstrated by the potentially huge growth in the Internet of Things (IoT) and Machine-to-Machine Communication (M2M), brings its own challenges for standardization and interoperability. No longer may a single standards body be responsible for an entire technology. Complex products and systems are often based on multiple standards from, for example, ETSI, the IETF, the IEEE or the ITU-T as well as on requirements published by industrial fora. Furthermore, interoperability problems may be compounded by the fact that standards are being used in contexts that the original specifiers did not foresee.

Good technical quality is essential in any standard. Ambiguities, errors, unclear requirements, conflicting options and other factors that could lead to non-interoperability must be reduced to a minimum.

THE SOLUTION

Within ETSI, it is the task of the Centre for Testing and Interoperability – CTI – to support the Institute's Technical Committees in the use of best practices for the specification and validation of base standards and the development of test specifications for key ETSI technologies. In doing this, the CTI has become a world leader in the use of specification languages, the application of validation, analysis and simulation techniques and in interoperability testing.

The CTI promotes best practices which encompass techniques which are pragmatic (including the validation of standards through interoperability events) as well as technically advanced (for example, using languages such as TTCN-3 to define test scenarios). These techniques have the added benefit of being applicable well beyond the world of standardization. Good practice incorporated into standardization can be carried through to proprietary development processes.



Experience proves that the application of these best practices does indeed help to develop interoperable standards which, in turn, lead to interoperable products.

2. Interoperability Overview

DEFINITION

There is no single definition of interoperability that will satisfy all readers. The following statement can be found at Wikipedia: *Interoperability is a property of a product or system, whose interfaces are completely understood, to work with other products or systems, present or future, without any restricted access or implementation.*

Interoperability is often thought of as little more than a testing activity. Rather, it should be regarded as a thread running through the entire standards development process and not as an isolated issue to be fixed at the end. Of course, testing is an important part of assuring interoperability but it is almost meaningless if the initial requirements gathering and the specification process do not consider interoperability as a fundamental objective.

INTEROPERABILITY IN STANDARDS

Building Interoperability into Standards

One of the aims of communications standardization is to ensure that implementations of a single standard (or set of standards) will interoperate without the need for complex proprietary interfacing equipment. In order to achieve this goal, it is necessary to consider interoperability right from the start. Specifications must anticipate all of the likely scenarios which could arise from the interchange of commands and data between two implementations from different manufacturers.

The interoperability of products implementing standards can only be guaranteed if:

- The specified protocols are robust and efficient
- The specified behaviour, data formats and encodings are clear and unambiguous
- Specifications are designed (rather than evolved) to be flexible, robust and predictable

Determining if Standards are Interoperable

Once a set of requirements has been identified and defined, it is important to validate that they do, in fact, offer an interoperable standardized solution. Many issues can be identified and rectified through continuous technical review, but an assurance of interoperability can only be reached through practical trials such as interoperability events, or Plugtests™.

ETSI Plugtests™ events complement other activities within the standardization process such as the development of conformance testing standards. The results of Plugtests™ events also provide valuable feedback to other international organizations and fora

Determining the Ability of Products to Interoperate

Testing is an important part of providing this guarantee of interoperability and there are three different types of test activity that should be considered:

1. **Conformance testing** involves connecting a device to a test system and operating a set of stringently defined tests. This ensures that a (single) product implements the requirements laid down in a standard correctly.
2. **Interoperability testing** involves connecting devices from different vendors and operating them in a variety of real-life scenarios. Often this will be done at so called interoperability events (or PlugtestsTM). For ETSI, the feedback from such events is extremely valuable in helping to validate the standards themselves. In addition there are obvious benefits gained by product developers from this type of testing.
3. **Interoperability testing with some conformance checking** ensures that two or more products interoperate and that they do so by exchanging information according to the applicable standards. This approach is often employed in an ETSI Plugtests™ event at various stages during the development cycle.

ETSI Plugtests™ *events bring together products implementing standardized protocols regardless of whether they are early prototypes or market-ready products. These events are an opportunity to test the capabilities of these products to interoperate with others while also providing a non-rigorous assessment of their level of conformance to the standards. Although Plugtests events are of considerable value to the product developers who take part in them, ETSI also benefits by having its standards validated due to the extensive testing that takes place.*

Conformance and interoperability testing are complementary to each other. Each has its strengths and weaknesses but together they can provide the best assurance that products will interoperate.

Certification ensures that a product can legitimately claim to have implemented a standard correctly. It can involve:

- Comprehensive conformance testing, or
- Comprehensive interoperability testing , or
- Both conformance and interoperability testing.

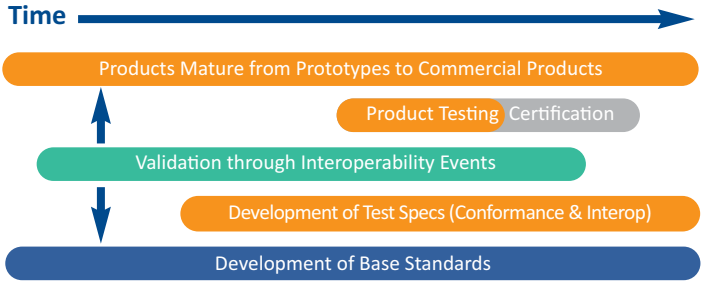
While ETSI does not operate its own certification programs, ETSI test suites may often used by third parties for certification.

Overview of the ETSI Approach

Despite its rather daunting name, a methodology is no more than a toolkit of methods, procedures and techniques that can be applied within a particular discipline. An interoperability methodology must include such tools to be used in activities which range from the collection and formulation of requirements right through to the specification and validation of test suites.

The process is not sequential as products and standards typically evolve in parallel with feedback flowing in both directions. Validating a standard and developing appropriate test specifications should be interleaved with the evolution of the standard itself. The CTI works with the ETSI Technical Bodies to set up test programs, and to develop test specifications which, like any other standards, can be freely used by industry. However, the actual testing of a product is, for the most part, outside ETSI's responsibilities.

Relationship between Standards, Validation & Testing



Interoperability Framework

A methodology is likely to offer a number of generic tools with various options which can be modified to suit a variety of broadly similar application areas. In order to optimise the methodology for a specific project or class of projects, it is useful to construct a Framework. This closes options, selects specific methods and provides additional information on how to use the selected methods. It may also specify procedures to help link the methods into a coherent overall process.

An Interoperability Framework should be well documented and should include guidance on the following:

Identification of specific methods, techniques and tools to be used for:

- *developing the base standard(s)*
  - Collecting and classifying base requirements
  - Specifying implementation requirements within the standards
  - Validating the completeness and correctness of each standard
- *cataloguing the testable requirements*
- *specifying conformance and interoperability test suites*
  - Defining the Test Suite Structure (TSS)
  - Using text and/or tables to specify Test Purposes (TPs)
  - Using tables and/or TTCN-3 to specify individual tests

Naming conventions, for example:

- *requirements identifiers within the catalogue*
- *test component identifiers*
- *test Purpose identifiers*
- *programming identifiers within TTCN-3*

Library conventions:

- *storage and management of textual items*
- *structure and management of requirements catalogues*
- *structure and management of TTCN-3 programming libraries*

Planning for testing and validation, such as defining:

- *which particular validation methods should be used within the overall development cycle?*
- *Will it be beneficial to include Plugtests™ events as part of the validation process?*
- *At what point in the standards development schedule should the specification of test suites commence?*

THE ETSI CENTRE FOR TESTING AND INTEROPERABILITY

CTI Services

ETSI's Centre for Testing and Interoperability (CTI) provides hands-on expertise and support to the Institute's Technical Committees and the Third Generation Partnership Project (3GPP™).

The CTI's principal task is the planning and development of conformance and interoperability test specifications. While of direct use to the ETSI Membership, many of these specifications are also used in external certification schemes such as those of the Global Certification Forum (GCF) and the DECT™ Forum.

In order to ensure the quality of their test specifications, CTI validates each one either in-house in collaboration with commercial test laboratories and recognised test tool suppliers or externally among the ETSI Membership.

The CTI also organises and manages ETSI Plugtests™ and specialises in the organisation of interoperability events for any ICT standard or set of standards. Since the late 1990s, ETSI has organised an average of 12 such events every year, worldwide.

Supporting Public Policy

The CTI plays a significant role in supporting ETSI Members by promoting and leading the production of conformance test specifications and by organising interoperability events in response to EU mandates and other public policies (especially the EC ICT Standardization Work Programme).

These events and ETSI's high quality test specifications are accepted at the international level and thus benefit individual users, industry and the public sector worldwide.
Many interoperability events and test specification activities are supported by the European Commission.

Plugtests Event In Action

3. Base Standard Development

REQUIREMENTS CAPTURE

There is no single method specified for the identification and classification of the requirements to be satisfied by a base communications standard (or set of standards). Most commonly within ETSI, however, consensus on the objectives and requirements is reached through the open discussion of proposals from member organizations. Whatever method is adopted, for a system design to be successful it necessary to be able to show the relationships which exist between objectives, requirements and the system design.

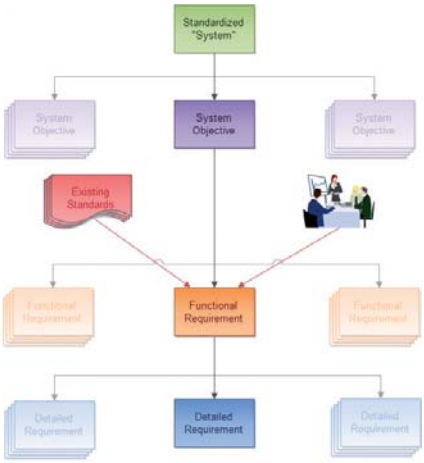
The distinction between an objective and a requirement is an important one to make:

- ***An objective is the expression of what a standardized system should be able to do in very broad terms***
- ***A requirement is a more detailed specification of how an objective is achieved***

Derived from the objectives and subdivided as:

- *Functional Requirements* identify the major functions to be used to realize the system Objectives. They are specified at a level which gives an indication of the broad behaviour expected of the asset, generally from the user's perspective.
- *Detailed Requirements,* as their name implies, specify a much lower-level of behaviour which would, for example, be implementable in a product and testable at a communications interface.

The diagram shows how Functional Requirements can be extracted from existing specifications and from other input and that they are combined to achieve the Objectives of the target system. Each Functional Requirement is realized by a number of Detailed Requirements.



SPECIFICATION OF REQUIREMENTS

The interoperability of implementations from different manufacturers is the fundamental purpose of standardization. This can only be achieved in a "top-down" approach to the development of standards, starting with Objectives, deriving Functional Requirements and then Detailed Requirements.

Recommendation I.130 from ITU-T describes a 3stage process which, although originally intended for use with protocol and service standards, specifies an underlying "top-down" method which can be applied to any type of standard. The three design stages that I.130 identifies are:

1. Requirements specification
2. Functional model and information flows
3. Detailed design

Stage 1 comprises the specification of Functional Requirements; Stage 3 comprises the specification of Detailed Requirements and Stage 2 describes the intermediate process for deriving Detailed Requirements from Functional requirements.

SPECIFYING SYSTEM OBJECTIVES

Systems Objectives should reflect the high-level goals of the system in providing the expected functionality to users. This means that they should be expressed:

- in terms of the expected behaviour of an implementation of the target standard(s). For example:
  - an IPv6 router
  - an IMS CSCF
- as desires rather than strict requirements:
- as abstract, system-wide capabilities. For example:
  - an IPsec host should be able to ensure that data transmitted to another IPsec device cannot be revealed to a third (unwanted) party.

VALIDATING OBJECTIVES

All system objectives should be assessed to ensure that they meet the 4 criteria of being:

- **Realistic**

The objective does not make unjustifiable demands on the target system. For example, in a secure environment it would be unrealistic to set an objective that all users should be able to view the secret passwords of all other users;

- **Achievable**

It should be possible to meet the objective within the bounds of current or emerging technology without unreasonable cost;

- **Measurable (i.e., testable)**

Once an objective has been met, it should be possible to view or otherwise validate its effect on the target system either directly or indirectly;

- **Relevant**

The objective should be directly related to the expected functionality of the target system and its environment.

SPECIFYING FUNCTIONAL REQUIREMENTS

No simple or automatic method exists for the extraction of Functional Requirements from a System Objective. In most cases it is a matter of asking questions such as *"How can this objective be achieved?"* or *"What behaviour is necessary to achieve this objective?"*

The specification of Functional Requirements should be limited to the following:

- Only those requirements that contribute to the achievement of the system objectives
  - If a new Functional Requirement is adjudged to specify essential behaviour but cannot be shown to contribute to the achievement of any of the System Objectives, a review of the Objectives should be undertaken in order to identify any additional objective(s)

- Only those requirements that are likely to have an impact on the communications interfaces, services or protocols of an implementation of the target standard(s)

A Functional Requirement should be expressed in a form that indicates whether the requirement is:

• **Mandatory:**
Uses, for example, the verb "shall"

"An implementation shall authenticate a user by means of a username and password"

• **Recommended:**
Uses, for example, the verb "should"

"An implementation should ensure that a user provides a valid identification prior to the start of a communications session"

• **Optional:**
Uses, for example, the verb "may"

"An implementation may use the NULL encryption algorithm to provide authentication and integrity if confidentiality is not a necessity"

Although, for the sake of consensus, it may seem attractive to include options and recommendations in a standard, the more they are used, the less likely it becomes that implementations will interoperate. A product that conforms to a standard that includes only mandatory requirements is almost certain to interoperate with other similar products. If it is essential to include an optional requirement within a standard, it should be expressed with a clear indication of the criteria which must be met if the option is to be selected.

"An implementation may include its IP address in an ERROR message if the Error Type is 'Onward Connection Unavailable'. It shall be omitted for all other Error Types"

SPECIFYING DETAILED REQUIREMENTS

Detailed Requirements specify the individual characteristics or elements of behaviour that a system must implement if it is to fully achieve the associated System Objectives. In many cases, the Functional Requirements will identify that Detailed Requirements can be derived directly from existing standards and international specifications. However, if no such specification exist, it may be necessary to define completely new requirements.

The development of Detailed Requirements should begin with a review of the Functional Requirements to determine how each of these can be broken down into lower level elements of behaviour. The use of graphical techniques such as Message Sequence Charts, UML activity diagrams or SDL process charts, to model behaviour can be very effective in identifying the Detailed Requirements which are the components of Functional Requirements.

The process of decomposing Functional Requirements should ensure that the resultant Detailed Requirements are atomic, i.e., they specify single characteristics or single elements of service or protocol behaviour.

Each Detailed Requirement should consist of the following:

- **an optional precondition which indicates the circumstances or context that must be established before the requirement becomes valid**

" If an ITS implementation supports the GeoAdhoc router functionality, ... "

- ***a stimulus defining the action which causes the security system to initiate a visible (measurable) response.***

" … a GeoAdhoc router receives a LS request packet… "

- ***a response defining the behaviour of the implementation on receiving the defined stimulus.***

" ... the GeoAdhoc router shall execute duplicate packet detection. "

There is no strict rule governing the order in which the precondition, stimulus and response should appear in an individual requirement. They should be arranged in such a way that the overall requirement is unambiguous and easy to read. The examples above could, therefore, be combined into a single detailed security requirement, thus:

" If an ITS implementation supports the GeoAdhoc router functionality,
and it receives a LS request packet, it shall execute duplicate packet detection"

4. Base Standard Validation

VALIDATING STANDARDIZED REQUIREMENTS

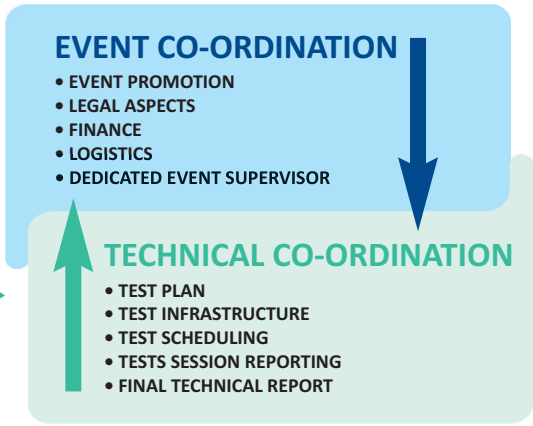
Once a set of requirements has been identified and defined, it is important to validate that they do, in fact, realize the specified objectives. There are a number of methods available for doing this but the three most effective are:

- **Structured Walk Through (Design Review):**
  - requirements are reviewed in detail by a group of experts to assess their behaviour both singly and, more importantly, in combination
  - validation is unlikely to be exhaustive
  - low cost
  - requires no additional modelling or emulation software tools
  - requires no prototype implementation equipment or infrastructure
- **ETSI Plugtests™ Event:**
  - an organized event at which implementations of the base standard(s) are interconnected and tested for interoperability and conformance
  - provides validation of both the base standard(s) and the implementations
  - scope of validation can be exhaustive or focused as necessary
  - requires products (prototypes or commercial products) and interconnection facilities
- **Modelling and Simulation:**
  - requirements are modelled using a formal language such as UML or SDL
  - models can be executed to simulate the specified behaviour
  - scope of validation can be exhaustive or focused as necessary
  - requires special modelling software tools and the skills to use them effectively.

VALIDATION OF BASE SPECIFICATIONS THROUGH INTEROPERABILITY EVENTS

ETSI has organized numerous interoperability events, or Plugtests™, across a wide range of different technologies. These events, which may comprise just a few participants or many hundreds, provide a focus for engineers to bring their equipment to a neutral environment (possibly distributed) where they can execute real-life test scenarios with equipment from other manufacturers. Plugtests™ events reduce time to market and speed up the standardization process.

HOW ARE PLUGTESTS ORGANISED?



5. Test Specification Development

At an appropriate point in the development of the base standard(s) it is possible to commence the test specification process. The exact point when this can occur differs from project to project but it is unlikely to be effective to start development of the test specifications before work has started on the Detailed Requirements.

Test Specifications generally comprise a reasonably rigid collection of documents, the structure of which is similar for both conformance and interoperability tests:

Document Type	Conformance Test Specification	Interoperability Test Specification
Collection of Testable Requirements	Implementation Conformance Statement Requirements Catalogue	Interoperable Features Statement Requirements Catalogue
Testing Overview	Test System Structure and Test Purposes	Test System Structure and Test Purposes
Intermediate Test Design	Test Descriptions	n/a
Test Suite	Automated Test Cases	Test Descriptions Automated Test Cases

COLLECTION OF TESTABLE REQUIREMENTS

Implementation Conformance Statement

An Implementation Conformance Statement (ICS) is a standardized proforma which collects together the Detailed Requirements in a base standard into a set of tabulated questions (to be answered by an implementer) with a constrained set of possible answers.

An ICS contains two types of questions:

• **Questions to be answered by either "YES" or "NO"**

These are related to whether a feature (whether an overall function or an element within a protocol message) has been implemented or not.

• **Questions on numerical values implemented**

These relate to timers, message lengths, frequencies and other similar parameters

The values permitted in the answers are constrained by the ranges specified in the base standard.

Although primarily intended for use by a product developer to ensure that all requirements of an implemented standard have been met, the ICS is also invaluable to test specifiers in determining:

- which requirements should be tested
- how each requirement should be tested
- what constitutes a test "Pass" or "Fail"

As a general rule, it is the base standard developers who are expected to produce the ICS. It is not unusual, however, for the test writers to collate the ICS to allow them to proceed with developing the test specification.

Interoperable Features Statement

The structure of an Interoperable Features Statement (IFS) is similar to that of an ICS. Its purpose is to identify the functions specified in the base standard(s) which an implementation should support, those which are optional and those which are conditional on the support of other functions. Although not strictly part of the interoperability test suite, the IFS helps to provide a structure to the suite of tests which will subsequently be developed.

As with the ICS, an IFS can be also be used by a manufacturer as a proforma for identifying which functions an EUT will support when interoperating with similar equipment from other manufacturers.

The ideal starting point in the development of an IFS is the ICS which should clearly identify the options and conditions which apply to the protocol to be tested. Like the ICS, the IFS should be considered part of the base protocol specification and not a testing document.

$Q_{\#}$	ICS/IFS Question	Reference	Status	Support
Q_1	Support of Feature F1	[x] Clause a	Optional	Yes/No
Q_2	Support of Feature F2	[x] Clause b	Mandatory	Yes/No
:	:	:	:	:
Q_n	Support of Message M1	[y] Clause c	Conditional (Note)	Yes/No
:	:	:	:	:
Q_k	Support of Message Parameter P1	[y] Clause d	Optional	Yes/No
Note: if Q1 = "Yes" then Mandatory else Optional				

Requirements Catalogue

Both the ICS and the IFS are good vehicles for the collection of testable requirements from a single base standard or even a coordinated set of specifications from a single standards organization. However, many of today's technologies are standardized as groups of related but nevertheless disjoint specifications from a variety of sources. This is particularly true of IP standardization. Building a coherent set of test specifications from disperse requirements sources can be simplified by gathering the requirements together into a single catalogue.

A Requirements Catalogue lists all implementation requirements from the various sources and organizes them into an appropriate structure. In most cases, creating a tree structure based upon functionality is a valid approach to structuring the requirements. Each node of the tree represents a specified function. Specific requirements are then associated with the relevant function node.

Details of each requirement can be entered in the Requirements Catalogue which is best implemented as a database.

As an example, each requirement in a particular catalogue could have the following information present:

Information Item	Notes
The (functional) group to which the requirement belongs	
A unique requirement identifier	
The identifier(s) of the test purpose(s) written for this requirement	There may be no TPs associated with the requirement
The requirement in text	• A direct quote from the source text. • Some simplification may be necessary in order to improve readability • In no event should the substance of the source's requirement be changed in transcribing it to the catalogue
A reference to the source of the requirement	Should indicate document, clause and, if necessary, paragraph to ease cross-referencing
The requirement type	Mandatory Optional Conditional

TEST SUITE STRUCTURE AND TEST PURPOSES

Test Suite Structure

Test Suite Structure (TSS) groups are arbitrarily chosen according to natural divisions in the base specification(s) and should be meaningful within the context of the testing project. The following criteria are examples which could be used either singly or in combination as the basis for TSS grouping:

- **The architecture of the testing configuration**

All test purposes explicitly requiring an upper tester are collected into a single group.

- **System behaviour**

Test purposes related to "normal" behaviour are separated from those related to "exceptional" behaviour.

- **Table of Contents from the base protocol specification**

Test purposes related to each described function within the base specification are grouped together according to those functions.

In most cases it is useful to base TSS grouping on a combination of criteria in order to avoid a structure that is "flat" and, therefore, not particularly meaningful.

Test Purposes

A Test Purpose (TP) should be written for each potential test of each identified requirement remembering that:

- a requirement may need more than one TP to ensure that it is fully tested
- some requirements may not be testable and so will have no TPs
- it may be possible to test more than one requirement with a single TP.

A TP describes in broad terms what the goal of a particular test should be. Each TP fits into one of the TSS groups and when viewed together with the TSS and all other TPs gives a clear picture of the scope (coverage) of the complete test specification.

A TP defines:

- the initial conditions to be established before testing can take place
- the status that must be established within the equipment to be tested before testing can proceed
- what is to be tested (NOT how it is to be tested)
- the criteria upon which verdicts can be assigned.

There is considerable benefit to be gained from having all Test Purposes written in a similar and consistent way. With a few simple rules established on a project-wide basis, this can be achieved in a number of different ways:

1. Free text with an underlying loose structure

This allows the TP writer considerable freedom of expression while also constraining the structure of the text and the use of particular English words and phrases in order to make each TP easier to read.

TP id	*:*	*TP/GEONW/PON/GUC/BV/01*
Title	*:*	*Test that the reception of a unicast packet over the upper Gn SAP triggers the origination of a GeoUnicast packet*
Role	*:*	*ITS Station*
Config	*:*	*CF01*
TC ref	*:*	*TC/GEONW/PON/GUC/BV/01*

Establish the IUT in the "initial" state.

Ensure that the IUT has received at least one "Beacon Information" packet from ItsNodeB.

When the IUT is requested by an upper layer to send a GeoUnicast packet to ItsNodeB, then it transmits a GeoNetworking packet with the "HT" field in the common header set to the value 2 (GEOUNICAST) and the "DEPV" field in the GeoUnicast Extended Header set to the position indicated by the SOPV value in the received "Beacon Information" packet.

2. Tabulated text

A fixed table structure removes some freedom of expression from the writer but ensures that all the required information is included in the TP or, at least, makes it easy to identify where information has been omitted.

Test Purpose	*TP/GEONW/PON/GUC/BV/01*		
Title	*Test that the reception of a unicast packet over the upper Gn SAP triggers the origination of a GeoUnicast packet*		
Role	*ITS Station*	***Config***	*CF01*
Initial conditions	*IUT established in the "initial" state* *IUT has received at least one "Beacon Information" packet*		
Stimulus	*IUT is requested by an upper layer to send a GeoUnicast packet to ItsNodeB*		
Response	*IUT transmits a GeoNetworking packet with the "HT" field in the common header set to the value 2 (GEOUNICAST) and the "DEPV" field in the GeoUnicast Extended Header set to the position indicated by the SOPV value in the received "Beacon Information" packet.*		

3. The standardized TP notation, TPLan

A simple, structured notation has been developed by ETSI for the expression of TPs. It combines the structure of tables with the readability of free text. Most importantly, the syntax of TPs written in TPLan can be checked with automatic tools.

```
TP Id     :  TP /GEONW/PON/GUC/BV/01
summary   :  'Test that the reception of a unicast packet over the
             upper Gn SAP triggers the origination of a GeoUnicast packet'
RQ ref    :  TS 102636-4-1 9.2.3.3, 9.3.4.2
Config    :  CF01
TC ref    :  TP /GEONW/PON/GUC/BV/01

with {      IUT established in the Initial state
       and IUT having received a BeaconInformation from ItsNodeB
     }
ensure that
     {  when {  IUT receives a request from an Upper_Layer
                     to send a GeoUnicast_Packet to ItsNodeB
             }
        then {  IUT sends a GeoNetworking_Packet
                     containing a CommonHeader
                       containing HT set to 2 'GEOUNICAST'
                   and containing a GeoUnicast_ExtendedHeader
                       containing DEPV set to SOPV from BeaconInformation
             }
      }
```

TEST DESCRIPTIONS

Test Descriptions specify the sequence of actions required to realize the verdict identified in the TP and are primarily intended for use in interoperability test specifications. However, in some instances, particularly where there is a considerable difference in complexity between the TPs and the TCs, it is worthwhile adding Test Descriptions as an extra design stage in a conformance test specification.

A test description should include as a minimum:

- a unique test description identifier
- a concise summary of the test which should reflect the purpose of the test and enable readers to easily distinguish this test from any other test in the document
- a list of references to the base specification section(s), use case(s), requirement(s), TP(s) which are either used in the test or define the functionality being tested
- a list of features and capabilities which are required to be supported by the SUT in order to execute this test (e.g. if this list contains an optional feature to be supported, then the test is optional)
- a list of all required equipment for testing and possibly also including a (reference to) an illustration (or a reference to it) of a test architecture or test configuration
- a list of test specific pre-conditions that need to be met before the test sequence can commence
- an ordered list of manual or automated operations and observations.

Interoperability Test Descriptions

As interoperability testing most often involves the activation and observation of user functions, it is reasonable for test cases to be specified as series of steps to be performed by human test drivers. Such test cases are more often referred to as Test Descriptions

In some instances, although not necessarily all, it is useful to be able to specify some pre-conditions to a test. This often takes the form of instructions for configuring the network and equipment to ensure that the Test Purpose is met fully.

Configure devices to communicate using SIP with G.711 µLaw codec.

The test steps themselves should be specified in a clear and unambiguous way but without placing unreasonable restrictions on how each step is performed. Clarity and precision are important to ensure that the step is followed exactly. The lack of restrictions is necessary if the test could apply to a range of different types of implementation.

Pick up User A's telephone handset and dial the number of User B

This is no less clear or unambiguous but it can be applied to any type of telephone.

At the end of each test case (and, where necessary, interspersed with the test steps) it is important to specify the criterion for assigning a verdict to the test case. This is probably best expressed as a question.

"Can speech from User B be heard and understood?"

Verdict criteria need to be specified as clearly and unambiguously as test steps and without restrictions. If a criterion is expressed as a question, it should be constructed in such a way that "Yes" and "No" are the only possible answers and it should be clear which result represents a "Pass" verdict and which represents a "Fail".

Both intermediate and final verdicts should be constructed in such a way that failure automatically implies failure of the overall test rather than a specific item of equipment. Intermediate verdicts should not be included simply to provide information. As an example, in an interoperability test suite for telephony functions, it would not be necessary to have an intermediate verdict "Is dial-tone present?" if dial-tone is intended to be generated locally. If, on the other hand, dial-tone should (or could) be generated by the remote end, such a verdict would be perfectly valid.

Although it is clear that a "Pass" verdict will always mean that, for a specific test, the connected devices interoperate correctly, it may not be the case that a "Fail" verdict implies that they do not. The interconnecting network equipment plays an essential role in almost all interoperability tests but is not usually included in the equipment being tested. A "Fail" verdict may be caused by a fault or unexpected behaviour in the network. Thus, each "Fail" verdict should be investigated thoroughly, possibly using monitoring equipment to determine its root cause before either validating the verdict as a true failure (if the root cause is within the tested devices) or retesting.

Both test steps and verdicts should be specified at the level appropriate to the functions to be tested. For example, if the purpose of an interoperability test suite is to test a telephony application where SIP is the underlying protocol, the test steps should specify actions and observations at the user terminal or agent

Answer incoming call
Is ringing tone heard?

If, however, the object is to establish the interoperability of two SIP protocol stacks, the tests should specify actions and observations possible at the application interfaces of the stacks

Cause SIP INVITE message to be sent
Was 180 Ringing received?

A good way to summarize test descriptions is to use a tabular format as illustrated in the example below.

	Test Description					
Identifier:	**TD_COR_8337_01**					
Summary:	IPv6 Router uses Router Advertisement to announce change of link-local address					
Test Purpose:	TP_COR_8337_01	**Reference:**	RQ_COR_8337_01	**Configuration:**	CF_022_I	
Pre-test	• EUT1 is configured as a router					
conditions:	• EUT1, EUT1 and EUT2 all subscribed to the same multicast group					
Step	**Test Sequence**					
	Cause EUT1 to assign a new link-local address to its interface to EUT1 and EUT2					
	Cause EUT1 to send an echo request to the Link-Local address of EUT1					
	Check: Protocol monitor shows that an echo request was sent from EUT1 to EUT1					
	Check: EUT1 receives an echo reply from EUT1					
	Cause EUT1 to send an echo request to the Link-Local address of EUT2					
	Check: Protocol monitor shows that an echo request was sent from EUT1 to EUT2					
	Check: EUT1 receives an echo reply from EUT2					
	Cause EUT1 to send an echo request to the new Link-Local address of EUT1					
	Check: Protocol monitor shows that an echo request was sent from EUT1 to EUT1					
	Check: EUT1 receives an echo reply from EUT1					
	Cause EUT2 to send an echo request to the new Link-Local address of EUT1					
	Check: Protocol monitor shows that an echo request was sent from EUT2 to EUT1					
	Check: EUT2 receives an echo reply from EUT1					
Observations						

Conformance Test Descriptions

In most instances the granularity of the Test Purposes and the information contained within them is sufficient that it is possible to specify conformance Test Cases in a language such as TTCN-3 directly from the TPs. If, however, an intermediate design stage is considered to be beneficial, it can take a similar form to the interoperability Test Description. The only significant difference between the two is that the test steps need to specify the behaviour of the protocol as observed at a test interface rather than as the functionality observed at the user interface.

TEST SUITE

Test Cases

A Test Case is a complete and independent specification of the actions required to achieve a specific test purpose. Each Test Case starts and ends in a stable operating state and can accommodate implementations running sequentially or concurrently. Test Cases are specified at a

level of detail that makes it possible for them to be written in a programming language, usually specific to testing, such as the advanced language TTCN-3 standardized by ETSI. Protocol speci fic run-time interfaces provide adaptors for timing and communication between a software test system, a particular processing platform and the system under test. Extensions to the TTCN-3 language support the specification of such interfaces.

TTCN-3 Benefits

Global Standard

- *standardized, modular language specifically designed for testing*
- *endorsed by ITU-T SG17 (Z.160 Series)*

Quality

- *usable in all phases of product development*
- *education and training costs can be rationalised*
- *maintenance of test suites (and products) is easier*
- *TTCN-3 Certificate programme for TTCN-3 Users*

Community

- *large and active user community*
- *multi vendor tool support*

Multi-Purpose Testing Language

- *all kinds of black-box testing (conformance, interoperability, performance, etc.)*
- *Independent of a specific application domain. Already used in:*
- *Mobile communications (LTE, WiMAX, 3G, TETRA, GSM)*
- *Broadband technologies (ATM, DSL)*
- *Middleware platforms*
- *Internet protocols (SIP, IMS, SIGTRAN and IPv6)*
- *Smart Cards*
- *Automotive (MOST, CAN)*
- *Intelligent Transport Systems (CAM, DENM, GeoNetworking)*
- *eDocument (ePassport, ePassportReader)*

The TTCN-3 Naming Conventions

The TTCN-3 core language contains several types of elements with different rules of usage. Applying naming conventions aims to enable the identification of the type when using specific identifiers according to the type of element.

For instance, a variable declared in a component has different scoping rules to those for a local variable declared in a test case. Identifiers of component variables are different from identifiers of local variables which makes it easier to recognize which type of variable the identifier belongs to.

Furthermore, applying naming conventions maintains the consistency of the TTCN-3 code across the test suites, and thus increases the readability for multiple users and eases the maintenance.

Example ETSI TTCN-3 Generic Naming Conventions

Language Element	Naming Convention	Prefix	Example Identifier
Constant	Use lower-case initial letter	c_	c_maxRetransmission

Test Cases Structure

In order to keep the readability, consistency and maintainability of the TTCN-3 code, test cases are implemented following a particular structure, as follows:

1. **Local variables:**
   declaration of the variables to be used within the test case
2. **Test control PICS:**
   one or more logical expressions that use module parameters (PICS or PIXIT) in order to decide whether or not to run the test case
3. **Initialization of component variables:**
   zero or more assignments that initialize or modify the value of component variables
4. **Test component configuration:**
   initializes or configures the components needed to run the test
5. **Test adapter configuration (optional):**
   configures the test adapter as needed to run the test
6. **Preamble:**
   contains the actions needed to bring the IUT into the desired state
7. **Test body:**
   contains the actions needed to run the test case. This part will set the verdict of the test
8. **Postamble:**
   contains the actions needed to bring the IUT into the initial state or the desired final state.

Test Synchronization

The synchronization of distributed TTCN-3 elements is an essential part of the development of a test suite. Consequently, a suitable and well-defined synchronization protocol should be used, even if this entails designing and developing one specifically for the test application. ETSI has developed a TTCN-3 library module which has been designed explicitly for the control and synchronization of distributed test systems.

TEST SUITE VALIDATION

In the same way that the quality of base standards can be assured by undertaking some form of validation of its contents, test specifications can also benefit from validation activities.

Test Suite Validation - Design Review

The formalized design review method of validation can be quite effective in validating the Test System Structure and the test purposes as they generally use a mixture of textual and graphical specification techniques. Such a review will easily highlight gaps in the test coverage of the base requirements as well as any unnecessary testing of unspecified functions. It will also validate that the general approach to testing, the test configurations and the individual test methods

implied by the test purposes are all valid.
Test Suites, however, cannot easily be validated by means of a design review as they are written in TTCN-3 or some other programming language. It is, of course, possible to partially validate the structure and coverage of a test suite by such a manual method but the correctness and effectiveness of a TTCN-3 test suite can only be validated by execution of the programming code in either a simulation or a controlled testbed.

Test Suite Validation - Simulation

Many TTCN-3 software tools offer some simulation facilities which can be used to validate a test suite. The capabilities vary between individual tools but at least one of the following methods is likely to be possible:

- **Back-to-back Validation:**
  - Execution of a compiled test suite within a development environment where the system under test is simulated as a "mirror" of the test suite.
  - No requirement for message CODEC.
  - Checks the logic of the ATS but does not validate its performance or resilience.
- **Message CODEC Validation:**
  - Validation of data encoding and decoding where the test suite and its CODECs are exercised in a real tester
  - One approach is to loop back all the data templates sent by the TTCN-3 test suite.
  - Another approach is to record traces from a real SUT and to feed the traces to the CODEC.

Simulation can be used very effectively to validate the underlying logic of a TTCN-3 test suite prior to more extensive testing in a test bed. However, each TTCN-3 tool will offer different levels of simulation and each is likely to operate in a different way. Consequently, advice should be sought from the supplier of the tool to determine its capabilities and the approach that should be taken.

Test Suite Validation - Testbed

Although the above methods can provide a reasonable level of validation of the test specification, it is only when it is implemented in a physical tester linked to an implementation of the base standard(s) that a true evaluation of the test suite can be made. These implementations of both the tester and the system under test could easily be prototypes. However, by configuring them together into a controlled hardware testbed, it is possible to validate not only the programmed behaviour of the test suite but also its ability to comply with the required response times and other performance-related characteristics.

The benefits of testbed validation are:

- Execution of a compiled test suite within a development environment.
- The system under test is a real implementation connected to the development environment.
  Adaptors and CODECs are essential.
  Checks the logic of the ATS as well as its resilience and, to a lesser extent, its performance.

Test Suite Validation - Quality Assurance Process

Each published test suite is accompanied with a validation report. The validation report provides information on the extent of validation of the test specifications (if any), and how the validation was performed. The following information should be included in the report:

- **Test Suite Validation Level:**

ETSI defines 3 validation levels defined.

- **TTCN-3 Edition:**

TTCN-3 is constantly maintained and and an indication of which version f TTCN-3 is used can be important information.

- **Identification of the TTCN-3 Tool:**

Name, version and all relevant details (settings, plugins etc) of the tool used to compile the TTCN-3 code.

- **Compilation status:**

Here the status (success, failure) of the compilation is provided. Failure reasons are categorized (types of errors).

- **TTCN-3 Quality:**

A critical issue for ETSI in developing test suites is the readability, consistency and maintainability of the TTCN-3 code. Automating the code analysis for coding guidelines (naming conventions etc) increases the level of confidence regarding code quality. It is for this reason that ETSI provides a quality checker tool, T3Q.

- **Test Platform Identification:**

Name, version and all relevant details of the test platform which integrates the TTCN-3 test suite.

- **IUT Identification:**

Name, version and all relevant details of the SUT used to validate the TTCN-3 test suite.

- **Validation Status of the test suite:**

A list showing which tests where validated.

6. Further Useful Reading

ETSI Publications:

EG 201 058	Methods for Testing and Specification (MTS); Implementation Conformance Statement (ICS) Proforma Style Guide.
EG 202 107	Methods for Testing and Specification (MTS); Planning for Validation and Testing in the Standards-Making Process.
EG 202 337	Methods for Testing and Specification (MTS); Internet Protocol Testing (IPT); Generic Approach to Interoperability Testing
EG 202 568	Methods for Testing and Specification (MTS); Internet Protocol Testing (IPT); Testing: Methodology and Framework
ES 201 873	Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3 (Multi-part series)
EG 202 553	Methods for Testing and Specification (MTS); TPLan: A Notation for Expressing Test Purposes
ETS 300 406	Methods for Testing and Specification (MTS); Protocol and Profile Conformance Testing Specifications; Standardization Methodology
TR 187 011	TISPAN; NGN Security; Application of ISO-15408-2 Requirements to ETSI Standards – Guide, Method and Application
ETSI White Paper #3	Achieving Technical Interoperability - the ETSI Approach

Other Publications:

An Introduction to TTCN-3	Authors: Colin Wilcock et al Published by Wiley Blackwell

Web Sites:

Plugtests	http://www.etsi.eu/WebSite/OurServices/plugtests/home.aspx
Making Better Standards	http://portal.etsi.org/mbs
TTCN-3	http://www.ttcn-3.org/StandardSuite.htm
TTCN-3 @ ITU-T	http://www.itu.int/en/ITU-T/studygroups/com17/Pages/ttcn references.aspx
T3Q and T3D	http://t3tools.informatik.uni-goettingen.de/trac/wiki/Download

Video:

ETSI Interoperability Events	http://vimeo.com/23255899

650 Route des Lucioles
06921 Sophia-Antipolis Cedex
France.
Tel: +33 (0)4 92 94 42 00
Fax: +33 (0)4 93 65 47 16

For further information, please visit :

www.etsi.org