World Class Standards

Solve
the Challenge of Interoperability!

Interoperability Best Practices

Sponsored by the European Commission

www.etsi.org

www.plugtests.org

Content

1. Market Drivers for Interoperability

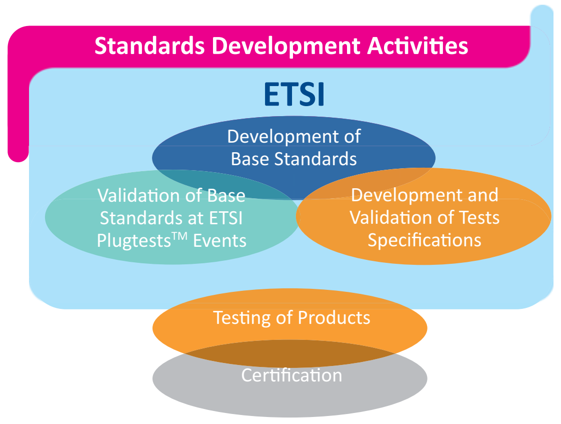
One of the underlying motives for the development of communications standards is to facilitate *interoperability* between products in a multi-vendor, multi-network and multi-service environment. It is these market demands that have ensured that interoperability has maintained, indeed increased, its prominence in standardization. Interoperability ensures that users have a much greater choice of products and that manufacturers can benefit from the economies of scale that a wider market makes possible. This brochure presents the ETSI approach to deliver interoperable standards through the application of best practices in specification, validation and testing.

> THE CHALLENGE

The trend towards a globally interconnected world, demonstrated by the potential huge growth in the *Internet of Things* (IoT) and *Machine2Machine Communication* (M2M), brings its own challenge 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 (e.g., from ETSI, IETF, IEEE, ITU-T) as well as on requirements laid down by industrial fora. Furthermore, potential interoperability problems may be compounded by the fact that standards are being used in contexts that the original specifiers did not foresee.
Even in individual standards, good technical quality is essential, ambiguities, errors, unclear requirements, conflicting options and other factors that could lead to non-interoperability must be reduced to a minimum.

> THE SOLUTION

In their efforts to deliver interoperable standards, ETSI's Technical Committees follow the principles of applying best practices for the specification and validation of base standards and the development of test specifications related to key ETSI technologies.
The techniques are pragmatic (for example, validation of standards through interoperability events) as well as technical (for example, the use of languages such as TTCN-3 to define test scenarios). They have the added benefit that these techniques are applicable well beyond the world of standardization. Good practice started in standardization can be carried through into proprietary development processes.


Standards Development Activities
ETSI
Development of Base Standards
Validation of Base Standards at ETSI Plugtests™ Events
Development and Validation of Tests Specifications
Testing of Products
Certification

The ETSI experience is that the application of these best practices does indeed help to develop interoperable standards, which in turn leads to interoperable products.

2. Interoperability Overview

> DEFINITION

There is no single definition of interoperability that will satisfy all readers. The following statement has been developed within ETSI over a number of years and seems to capture our vision of interoperability within standardization:

Interoperability can be considered to be the ability of two or more systems or components to exchange data and use information.

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 far less effective 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 (possibly from various different standards bodies) 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 messages and data between implementations from different manufacturers.

The interoperability of products implementing standards can only be guaranteed if:
• Interfaces and architectures are fully defined
• Specifications are designed (rather than built ad hoc)
• The specified protocols are robust, flexible and efficient
• The specified behaviour, data formats and encodings are clear and unambiguous
• The context in which the specifications are used is fully understood
• Specifications are well maintained

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. Obviously many issues can be identified and rectified through continuous technical review, but at some point the only way to tell is through practical trials such as interoperability events, or Plugtests™.
By incorporating Plugtests™ in the standardization process, ETSI complements its support to other testing activities (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 a guarantee of interoperability and there are three different types of test activity that should be considered:

1. **Conformance testing** ensures that a product correctly implements the standard and is able to exchange directives and information with another implementation using a known protocol or set of protocols
2. **Interoperability testing** is realized by connecting devices from different vendors and operating them, either manually or automatically, according to scenarios based on a protocol standard. The tests can be executed with or without some less comprehensive conformance checking. This is an approach often employed in an ETSI Plugtests™ event at various stages during the development cycle. In addition to the obvious benefits gained by product developers from this type of testing, feedback from such events is extremely valuable in helping to validate the standards themselves
3. **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

__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 as a by-product of the extensive testing that takes place.

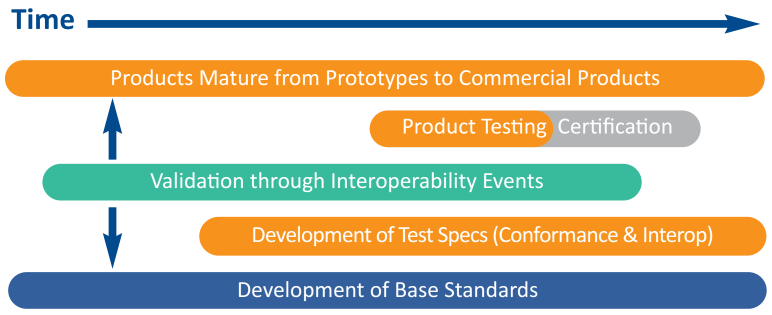
Conformance testing and interoperability testing are complimentary to each other. Each has its strengths and weaknesses but together they can provide the best assurance that products will interoperate.

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, of course, is not sequential. Typically, products and standards are evolving in parallel, with development feedback in both directions. The relationships between the various activities which constitute the development of a standard (or set of standards) are quite simple although there are some significant overlaps in both content and time. The activities involved in validating a standard and specifying appropriate test specifications should be interleaved with the development of the standard itself. All of these take place within ETSI but the actual testing of a product is, for the most part, beyond ETSI's responsibilities.

Relationship between Standards, Validation & Testing


Time
Products Mature from Prototypes to Commercial Products
Product Testing
Certification
Validation through Interoperability Events
Development of Test Specs (Conformance & Interop)
Development of Base Standards

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 languages such asTTCN-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 method should be used within the overall development cycle*
  - *>at which point in the standards development schedule the specification of test suites should commence*

> THE ETSI CENTRE FOR TESTING AND INTEROPERABILITY

Based on over 20 years of experience, the ETSI Centre for Testing and Interoperability (CTI) provides hands-on expertise and support to the ETSI Technical Organization and 3GPP for:

- Standards validation (focusing on Plugtests™ interoperability events)
- Planning and defining validation and testing strategies
- Development of test specifications (conformance and interoperability)
- Application of protocol specification techniques
- The application of best practices in interoperability and testing
- The application of best practices in protocol specification
- Advancement of methodologies and best practices, including TTCN-3

Plugtests Event In Action

3. Base Standard Development

> REQUIREMENTS CAPTURE

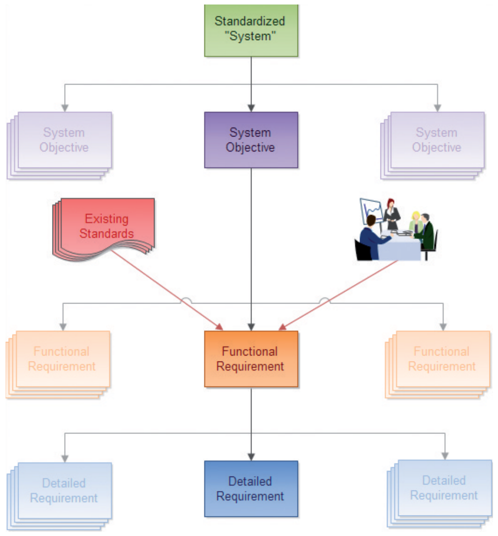
There is no single method specified for the identification and classification of the requirements to be satisfied by a base communications standard (or set of standards). Most commonly within ETSI, however, consensus on the objectives and requirements is reached through the open discussion of proposals 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. Functional requirements can be extracted from existing specifications and from other input and 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 3-stage 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*

Although there is no formalised place for the expression of standards system objectives, they should still be recorded and so it is worth considering what such an objective should look like.

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 or UML activity diagrams 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*

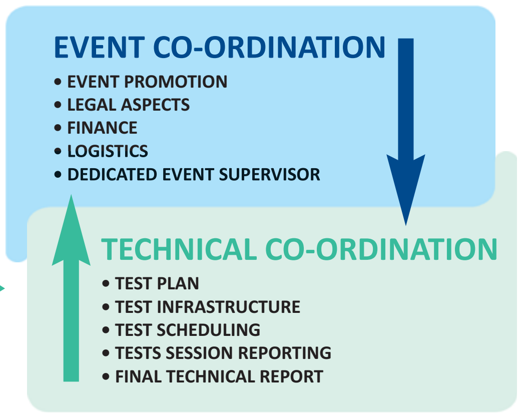
• ETSI Plugtests™ Event:
- *An organised event at which early prototype implementations of the base standard(s) are interconnected and tested for both interoperability and low-level conformance*
- *Provides validation of both the base standard(s) and the implementations*
- *Scope of validation can be 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

To date, ETSI has organized over 150 interoperability events, or Plugtests™, for at least 50 different technologies. These events, which may comprise just a few or many hundreds of participants, collect engineers and equipment in a neutral environment (possibly distributed) where they can execute a large variety of real-life testing scenarios in various combinations and with different equipments. Plugtests™ events reduce time to market and speed up the standardization process.

HOW DO WE ORGANISE PLUGTESTS?



4. 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 the 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 SYSTEM STRUCTURE AND TEST PURPOSES

Test System 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 test purposes
• It may be possible to test more than one requirement with a single test purpose

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

A test purpose 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.

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.

Example of Tabulated Text

TP Id	TP /GEONW/PON/GUC/BV/01
Test objective	Test that the reception of a unicast packet over upper Gn SAP triggers the origination of a GeoUnicast packet
Reference	TS 102636-4-1 9.2.3.3, 9.3.4.2
Config Id	CF01
PICS Selection	

Initial conditions

```
        with {
                the IUT being in the "initial state" and
                the IUT having received "Beacon information from ItsNodeB"
        }
```

Expected behaviour

```
    ensure that {
        when {
                IUT is requested to send ‘a GeoUnicast packet to ItsNodeB’
        }
        then {
                IUT sends a GeoNetworking packet
                        containing ‘a correctly formatted Common Header’
                                containing HT field
                                        indicating ‘value 2 (GEOUNICAST)’
                        containing GeoUnicast Extended Header
                                containing DEPV field
                                        indicating ‘same position as the SOPV
                                        value of the Beacon information received’
        }
    }
```

3. The standardized TP notation, TPLan

A simple, structured notation has been developed 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.

Test Descriptions

Test Descriptions (TD) specify the sequence of actions required to realize the verdict identified in the test purpose 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 test purposes and the Test Cases (TC), it is worthwhile adding test descriptions as an extra design stage in a conformance test specification.

Requirements on test description content

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 Bclear and unambiguous but it can only apply to a classical, physical telephone and not to a soft phone or even a mobile handset.

Initiate a new call at User A to the address of User B 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 ITS example below.

Interoperability Test Description			
Identifier	TD_GEO_ 01		
Summary	To check that the SUT exchanges correctly information using GeoUnicast forwarding mechanism		
Configuration	CF#1		
SUT	Vehicle 1, Vehicle 2 and Vehicle 3		
References	**Test Purpose**	**Specification Reference**	
	N/A	TS 102 636-4-1	
Pre-test conditions	Vehicle 1 and Vehicle 2 are within single-hop messaging distance Vehicle 2 and Vehicle 3 are within single-hop messaging distance Vehicle 1 and Vehicle 3 are not within single-hop messaging distance Vehicle 1, Vehicle 2 and Vehicle 3 are receiving beacon messages		
Test Sequence	**Step**	**Type**	**Description**
	1	stimulus	Vehicle 1 generates application information to be sent to Vehicle 3
	2	check	The IUT broadcasts a LS_REQUEST Geonetworking packet containing request field indicating GN_ADDR of vehicle 3 and IUT receives a LS_REPLY geonetworking packet originated by Vehicle 3
	3	check	Vehicle 1 selects vehicle 2 as next hop and sends the geounicast packet containing the information as payload, addressed to vehicle 3
	4	check	Vehicle 2 forwards the geounicast packet to vehicle 3
	5	verify	Vehicle 3 receives application information from vehicle 1

Note:

- A **stimulus** corresponds to an event that enforces the *Equipment Under Test* (EUT) to proceed with a specific protocol action, like sending a message for instance
- A **verify** consists of verifying that the EUT behaves according to the expected behaviour (for instance the EUT behaviour shows that it receives the expected message) at the user interface
- A **configure** corresponds to an action to modify the EUT configuration
- A **check** ensures the receipt of protocol messages at a test interface, with valid content. This "check" event type corresponds to the interoperability testing with conformance checking method

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 test purposes. 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 and endorsed by ITU-T as Z.16x series. Standardized interfaces allow easy integration of test cases with a test system. Extensions to the TTCN-3 language support the specification of such interfaces.

TTCN-3 feature set

- *Look and feel of a programming language*
- *Clearly separates interactions at different interfaces during a test with concurrent test components*
- *Data and signature templates with powerful matching mechanisms*
- *Handles test verdict mechanisms*
- *Test suite parameterization at runtime*
- *Scalable test campaign through test case selection mechanisms*
- *Harmonized with ASN.1, IDL and XML*
- *Intuitive and well-defined syntax and operational semantics*

The TTCN-3 Naming Conventions

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 than a local variable declared in a test case. Then identifiers of component variables are different from identifiers of local variables, in order 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

Besides these naming conventions, further recommendations are proposed with regard to:

• Structure of Data:

- data and other types should be defined in alphabetic order within TTCN-3 groups within the same TTCN-3 module
- all message types referenced in port type definitions and related to the same interface should be defined in the module, where these ports are defined

• Test Suite Modularization:

- usage of TTCN-3 code libraries
- organization of TTCN-3 code into different modules

• Log Statement:

- All TTCN-3 log statements must follow the following format:
  - three asterisks should be used to precede the log text
  - the TTCN-3 test case/function identifier in which the log statement is defined should follow
  - one of the categories INFO, WARNING, ERROR, PASS, FAIL, INCONC, TIMEOUT should follow
  - Free text should follow
  - And the log text should end with three asterisk

*log("*** f_sendMsg: INFO: Message has been sent ***")*

- any invocation of an external function must be followed by log statement
- each TTCN-3 setverdict statement that sets a test component verdict to INCONC or FAIL should be preceded by a log statement

TTCN-3 Code Documentation

Some of the most important rules are:

- All TTCN-3 testcase, altstep and function statements should be documented at least with @desc tag and @param for each parameter
- All TTCN-3 modulepar statements should be documented at least with @desc tag
- TTCN-3 modules should be documented at least @desc and @version tag

TTCN-3 benefits

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

• Independent of execution environment
- standardized test system interfaces for test execution, logging, and adaption

• Highest Flexibility in Designing and Maintaining Test Software
- support of automated and distributed testing
- reusability of test functionalities in different contexts

• 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.).

Already used in:
- Mobile communications (LTE, WiMAX, 3G, TETRA, GSM)
- Broadband technologies (ATM, DSL)
- Internet protocols (SIP, IMS, IPv6)
- Smart Cards
- Automotive (MOST, CAN)
- Intelligent Transport Systems
- eDocument (ePassport, ePassportReader)

Test Cases Structure

In order to keep the readability, consistency and maintainability of the TTCN-3 code, ITS test cases are implemented following a certain structure.

- Local variables: declaration of the variables to be used within the test case.
- 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
- Initialisation of component variables: contains zero or more assignments that initialize or modify the value of component variables
- Test component configuration: initialized or configures the components needed to run the test
- Test adapter configuration (optional): configures the test adapter as needed to run the test
- Preamble: contains the actions needed to bring the IUT into the desired state
- Test body: contains the actions needed to run the test case. This part will set the verdict of the test
- 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 specified 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 formalised 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.

- 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.

- Test Suite Validation Level: There are 3 validation level defined. This clause states the intended level of validation
- TTCN-3 Edition: TTCN-3 is constantly maintained and this clause indicates which version f TTCN-3 is used. This is particularly important information for the chosen TTCN-3 compiler
- 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. If T3Q has been used this clause indicates the settings of the T3Q configuration file
- Test platform identification: name, version and all relevant details of the test platform which integrates the TTCN-3 test suite
- SUT 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 N°3	Achieving Technical Interoperability - the ETSI Approach

Other Publications:

An Introduction to TTCN-3	Authors: Colin Wilcock et al Published by Wiley Blackwell ISBN 9780470012246

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

World Class Standards