Draft ETSI EG 2XX XXX V<0.9.1> (2009-07)

ETSI Guide

Methods for Testing and Specification (MTS); Automated Interoperability Testing; Methodology and Framework

Reference

DEG/MTS-00119

Keywords

Interoperability, testing, methodology

Contents

Intellectual Property Rights

Foreword

This ETSI Guide (EG) has been produced by ETSI Technical Committee Methods for Testing and Specification (MTS).

Introduction

The encouragement of the European Commission for the adoption and promotion of generic test frameworks of standards based on multiple stacks including middleware is providing evidence that successful testing and interoperability are key factors enabling the use of new technologies and providing all benefits attached to them, i.e., competitiveness, innovation, etc. However technologies are becoming more and more complex, collaborative, inter-dependant etc. and methodologies and approaches for ensuring interoperability need also to be innovative and take into account new factors such as the distribution of the components, the difficulty to access to components locally due to the distance or the embedded environment. This project intends to adapt solid and proven methods to these new challenges.

The current and future e-communication market can be described as a convergent multimedia market with an increasingly complex structure. Within the present competitive environment, the risk of non-interoperability is increasing because of small windows of opportunity due to fast evolution of technology, or the use of non-open standard. The main aim of standardization is to enable interoperability in a multi-vendor, multi-network, multi-service environment. The absence of interoperability must not be the reason why final services for which there is great demand do not come into being.

ETSI is very much aware of these developments and market demands. It knows what the inhibitors to interoperability are that can be encountered during the standards development process. A key part of this process is the development of test specifications for conformance and interoperability and the provision of validation services based on many years of experience.

Testing the conformance of every single entity (protocol layer) in a complex system seems often prohibitively expensive. On the other hand pure interoperability testing does not ensure that standard specifications are being adhered to between communicating entities. For example, a recent study ETSI demonstrated 90% of a given set of systems to interoperable but only a conformance 60% to the standard used to link these systems together. In the field, this would undoubtedly lead to interoperability problems. Over the past few years a combination of these two testing approaches is giving promising results at ETSI which we call interoperability testing with conformance checking. The basic idea is simple: end-to-end interoperability tests are executed while at the same time key reference points (interfaces) are checked to see that the message flow conforms to the flows mandated by standards. So far however this approach has mainly applied with manual interoperability testing.

The focus of this document is a generic methodology for the automated interoperability testing with conformance checking of complex distributed systems. The proof of concept of this methodology has beendone through a case study in the domain of IMS. However, also other technologies such as IPv6, Robust Header Compression (ROHC), Health Level 7 messaging, grid, WiMax, IPTV, WiMedia, Voice over IP (VoIP) for air traffic control, and smart cards have been analyzed and considered in the writing of this document.

1 Scope

The present document describes a methodology and framework for automated interoperability testing extending the best working practices presented EG 202 237 [1] to include automation aspects. It also considers application domains outside of internet protocol testing in that it proposes an approach for harmonising the understanding of generic testing architectures of distributed systems in terms of, for example, access to reference points, the establishment and maintenance of test synchronisation, and the assignment of test verdicts. In addition, the document outlines recommendations for the use of TTCN-3 [6] for this purpose.

2 References

References are either specific (identified by date of publication and/or edition number or version number) or non-specific.

1. For a specific reference, subsequent revisions do not apply.
2. Non-specific reference may be made only to a complete document or a part thereof and only in the following cases:
    - if it is accepted that it will be possible to use all future changes of the referenced document for the purposes of the referring document;
    - for informative references.

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

For online referenced documents, information sufficient to identify and locate the source shall be provided. Preferably, the primary source of the referenced document should be cited, in order to ensure traceability. Furthermore, the reference should, as far as possible, remain valid for the expected life of the document. The reference shall include the method of access to the referenced document and the full network address, with the same punctuation and use of upper case and lower case letters.

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

2.1 Normative references

The following referenced documents are indispensable for the application of the present document. For dated references, only the edition cited applies. For non-specific references, the latest edition of the referenced document (including any amendments) applies.

[1] ETSI EG 202 237: "Methods for Testing and Specification (MTS); Internet Protocol Testing (IPT); Generic approach to interoperability testing".

[2] Hans van der Veer, Anthony Wiles: "Achieving Technical Interoperability – the ETSI Approach", ETSI White Paper No.3, Sophia-Antipolis, France, 2008

[3] ETSI EG 202 568: "Methods for Testing and Specification (MTS); Internet Protocol Testing (IPT); Testing: Methodology and Framework"

[4] ISO/IEC 7948-1: "Information technology - Open Systems Interconnection – Basic Reference Model: The Basic Model".

[5] ISO/IEC 9646-1: "Information technology-Open Systems Interconnection-Conformance testing methodology and framework - Part 1: General concepts"

[6] ETSI ES 201 873-1: "Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Parts 1: TTCN-3 Core Language" (also published as ITU-T Recommendation series Z.140)

[7] ETSI ES 202 553: "Methods for Testing and Specification (MTS); TPLan: A notation for expressing Test Purposes".

[8] ETSI ES 201 873-10: "Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Parts 10: TTCN-3 Documentation Comment Specification" (also published as ITU-T Recommendation series Z.140)

[9] S. Schulz, "Test suite development with TTCN-3 libraries", International Journal on Software Tools for Technology Transfer, 10(4), 327-36, Springer, 2008.

[10] ETSI TR 1xx xxx, "Methods for Testing and Specification (MTS); Automated Interoperability Testing; Specific Architectures".

3 Definitions, symbols and abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in [1] and the following apply:

abnormal test termination: describes the result of executing an abstract test case after it has been prematurely terminated by the test system

abstract test case: a complete and independent specification of the action required to achieve a specific test purpose, defined at the level of abstraction of a particular abstract test method, starting and ending in a stable testing state (ISO/IEC 9646-1)

abstract test suite: a test suite composed of abstract test cases (ISO/IEC 9646-1 [5])

base specification: a specification of a protocol, abstract syntax, encoding rules, or information object

end-to-end testing: a testing approach focused on verifying the correct behaviour of a set of interconnected systems by stimulating and observing the system functionalities from the end user's point of view

executable test case: a realization of an abstract test case (ISO/IEC 9646-1 [5])

executable test suite: a test suite composed of executable test cases (ISO/IEC 9646-1 [5])

fail verdict: a test verdict given when the observed test outcome either demonstrates non-interoperability of equipment under test with respect to the end-to-end functionality on which the test case is focused, or demonstrates non-conformance with respect to at least one of the conformance requirement(s) on which the test purpose(s) associated with the test case is (are) focused, or contains at least one invalid test event, with respect to the relevant specifications

NOTE: The above definition extends the original definition for this term provided in ISO/IEC 9646-1 [5].

inconclusive verdict: a test verdict given when the observed test outcome is such that neither a pass nor a fail verdict can be given (ISO/IEC 9646-1 [5]).

pass verdict: A test verdict given when the observed test outcome gives evidence of all equipment under test interoperating for the end-to-end functionality on which the test case is focused, or conformance to the conformance requirement(s) on which the test purpose(s) of the test case is (are) focused, and when all the test events are valid with respect to the relevant specifications

NOTE: The above definition extends the original definition for this term provided in ISO/IEC 9646-1 [5].

test architecture: an abstract description of logical entities as well as their interfaces and connections involved in a test

test case: a generic, abstract or executable test case (ISO/IEC 9646-1 [5])

test configuration: a concrete instance of a test architecture defined on the basis of test components, ports and their connection

test description: an informal description of a test case usually written in English prose

test event: an indivisible unit of test specification at the level of abstraction of the specification (e.g. sending or receiving a single message) (ISO/IEC 9646-1 [5])

test log: a human readable record of information produced as a result of a test campaign which is sufficient to record the observed test outcomes and to verify the assignment of the test result (including test verdicts)

test oracle: a mechanism that determines a test verdict

test purpose: a description in English prose or TPLan [7] of a well defined objective of testing, focusing on either a specific end-to-end functionality, a single conformance requirement or a set of related conformance requirements as specified in the base specification (e.g. verifying the support of a specific value of a specific parameter)

NOTE: The above definition extends the original definition for this term provided in ISO/IEC 9646-1 [5].

test step: a named subdivision of a test case, constructed from test events and/or other test steps (ISO/IEC 9646-1 [5])

test verdict: a statement of "pass", "fail" or "inconclusive", specified in an abstract interoperability test case, concerning either the end-to-end interoperability of two or more Equipment Under Test (EUT) or conformance of an Implementation Under Test (IUT) with respect to that test case when it is executed.

NOTE: The above definition extends the original definition for this term provided in ISO/IEC 9646-1 [5].

t**est outcome:** the sequence of test events, together with associated data and/or parameter values, which occurred during test execution of a specific parameterised executable test case (ISO/IEC 9646-1 [5])

test report: document that presents test results and other information relevant to a test.

test result: test verdicts and associated information produced as a result of running a test case.

3.2 Abbreviations

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

ATS	Abstract Test Suite
E2E	End-to-End
ETS	Executable Test Suite
ETSI	European Telecommunications Standard Institute
EUT	Equipment Under Test
IFS	Interoperability Feature Statement
IPT	Internet Protocol Testing
IUT	Implementation Under Test
MTC	Main Test Component
MTS	Methods for Testing and Specification
OSI	Open Systems Interconnection
PCO	Point of Control and Observation
PDU	Protocol Data Unit
PIXIT	Protocol Implementation eXtra Information for Testing
SUT	System Under Test
TC	Test Case
TD	Test Description
TP	Test Purpose
TTCN-3	Testing and Test Control Notation
UE	User Equipment

4 Basic concepts & guidelines

4.1. Interoperability of distributed systems

Interoperability is a key factor in the widespread commercial success of any given technology in the telecommunication sector. Interoperability fosters diversity as well as competition in a market. Vendors can achieve interoperability of their products only if they agree and implement a common set of open standards. However, standardization does not necessarily lead to interoperability. Standards have to be engineered for interoperability.

Generally, interoperability can be defined as “the ability of two systems to interoperate using the same communication protocol” [1, 2]. In the case of distributed systems, this means that they provide its users a specified End-to-End (E2E) functionality. This implies that each component of a distributed system is able to exchange information with other components across standardized interfaces using the communication protocols and procedures specified in order to provide the required E2E functionality to the end user.

Based on the OSI layered model [4], two forms of interoperability can be distinguished as shown in Figure 1, i.e., horizontal and vertical interoperability. Horizontal interoperability refers to interoperability within the same protocol layer or entities belonging to the same service, whereas vertical interoperability refers to the interoperability of entities belonging to different protocol layers or services. This document focuses mainly on the automation of horizontal interoperability assessment.

4.2. Interoperability testing of distributed systems

Systems that implement a set of standards need to be assessed for their interoperability with other systems. It is also necessary to demonstrate that they conform to these standards. At the standardization level, this in turn can be facilitated with the availability of open and validated test specifications.

Interoperability testing is the most intuitive way of confirming that two or more systems work together. A number of standardization organizations use interoperability testing events as a means to of raising the status of a specification to the level of a standard. ETSI uses interoperability testing events, so called Plugtests™, to provide feedback to technical bodies on the maturity of a given technology and its underlying standards. Such events can only be successful if testing is based upon an agreed set of interoperability tests.

The purpose of interoperability test specifications is to define the procedures necessary to prove that the E2E functionality between communicating systems which have been implemented based on the same specification, e.g., a set of standards. In the context of interoperability testing of distributed systems each system is called an Equipment Under Test (EUT) and the collection of all EUTs is called the System Under Test (SUT) as defined in ETSI EG 202 237 [1]. In order to verify the correctness of the protocol procedures and to provide a basis for fault analysis, interoperability test specifications can be combined with supplementary conformance checks when assessing E2E functionality. Such conformance checks provide further information regarding the correctness of the communication during E2E testing. NOTE: Conformance checks at the interfaces connecting different systems can be a useful source of information when troubleshooting in the case of a test case execution failure.

In addition, [1] describes the basic concepts of interoperability testing, a means of describing test architectures and a process for developing and executing interoperability test specifications . Figure 2 depicts an example interoperability test setup combining E2E interoperability tests with conformance checks. The present document extends the basic concepts and generic approach to interoperability testing defined in EG 202 237 to cover aspects related to the testing of distributed systems and the automation of interoperability testing.

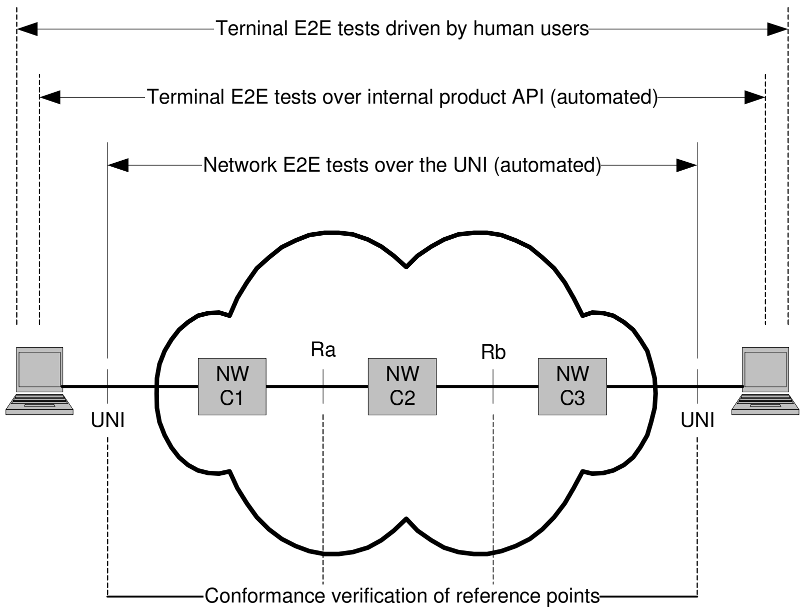
Terninal E2E tests driven by human users
Terminal E2E tests over internal product API (automated)
Network E2E tests over the UNI (automated)
Ra
Rb
NW C1
NW C2
NW C3
UNI
UNI
Conformance verification of reference points

Figure 2: Example of interoperability testing with conformance checking

4.3. Verdicts

Automated interoperability testing of distributed systems produces a large amount of information on several interfaces. In order to manage this information and to speed up troubleshooting, two verdicts should be managed independently:

- an E2E interoperability verdict which is based on the observation of SUT behaviour at its end points and should assume the following values:
  - **pass**: the observed test outcome demonstrates that all EUTs interoperate according to the end-to-end functionality required by the test;
  - **fail**: the observed test outcome demonstrates that some or all of the EUTs do not interoperate according to the end-to-end functionality required by the test
  - **inconclusive**: the observed test outcome is such that neither a pass nor a fail verdict can be given
- a conformance verdict which is based on observation of protocol procedures and messages exchanged at interfaces between EUTs as shown in Figure 2 and should assume the following values:
  - **pass**: the observed test outcome demonstrates conformance to the normative requirement(s) tested by the associated test case;;
  - **fail**: the observed test outcome demonstrates non-conformance to at least one of the to the normative requirement(s) tested by the associated test case;;
  - **inconclusive**: the observed test outcome is such that neither a pass nor a fail verdict can be given.

Table 1 shows different possible scenarios in verdict observations as well as their potential source of error. Note that verdicts alone do not allow direct identification of the cause of a test failure. Resolution of test failures is generally not trivial and may requires troubleshooting.

Table 1: Example interoperability verdict scenarios

End-to-End Verdict	Conformance verdict	Interpretation
Pass	Pass	EUTs have been interoperating and communicating according to the standard.
Pass	Fail	EUTs have been interoperating but not communicating according to the standard. This may be due to a problem in the base standard, the test system, or one of the EUTs.
Pass or Fail	Inconclusive	EUTs have been interoperating or not, all captured communication is according to the standard, but one interface with conformance checks has not been available for evaluation
Fail	Fail	EUTs have not been interoperating and not communicating according to the standard. This may be due to a problem in the standard, the test system, or one of the EUTs.

Each test step in an interoperability test may lead to intermediate verdict. Inserting checkpoints in a test to capture such verdicts for each significant test event can help to speed up the evaluation of the test execution results.

4.4. Automation

Reducing time to market for a new product, service, or architecture is a strategic requirement in a competitive environment, where it is important to introduce new services and technologies before or, at least, not after competitors. Automation can significantly reduce the time for testing and avoids repetitive manual activity which is prone to error. Consequently, it can lead to a considerable cost savings in terms of human expert resource as well as test bed usage, i.e., test equipment and EUT resource, acceptance costs both for suppliers and for customers, as well as manual interaction related to test execution, the analysis of trace and message contents and reporting.

A typical application of automated testing systems is related to regression testing. In this scenario, testing specialists are usually involved in repeating several time the same test suite consisting in a large number of tests. Nevertheless automation can also help in settings such as interoperability events, e.g., to analyze interoperability execution traces of the same test for different vendor pairings.

A general prerequisite for test automation is the availability and accessibility of interfaces used by the test specification to stimulate, observe or monitor the System Under Test (SUT).

4.4.1 Limitations

A cost/benefit analysis carried out prior to the development of test specifications can identify limitations and assist in determining a suitable degree of automation for interoperability test cases. This analysis should consider the topology of the distributed system, the protocols involved and their complexity, the stability of the specifications , and the resource required to develop a test system in terms of specialist manpower and hardware and software devices required .

The degree of test automation is often a compromise between testing requirements and feasibility. It may not be profitable, for example, to automate the entire testing process since the resources required to implement all or part of the tests may be prohibitive. In addition, some interfaces may require significantly less effort to assess manually.

4.4.2 Degree of automation

The following types of interoperability testing activity may be automated:

- configuration of EUTs and their interconnecting network;
- monitoring of relevant interfaces between EUTs;
- validation of EUT communication;
- simulation or emulation of (external) equipment;
- operation of all equipment involved in an interoperability test, e.g., EUTs;

- computation of test verdicts;
- test execution;
- generation of test reports.

The ability to automate these activities, i.e., the possible degree of automation, depends on the limitations described in the clause 4.4.1. The desired degree of automation will depend on the context in which the resultant tests are expected to be executed. For example, if the suite of tests is to form the basis of an interoperability event, it may not be desirable to emulate user equipment if the complexity of its interface to the EUT is such that there is a significant chance of introducing interoperability problems in the emulation which will mask or overwhelm any errors that might exist in the EUT itself. In such a case, one the use of commercial equipment (whose operation could possibly be automated) would be preferable.

In practice, complete automation of interoperability testing is rarely achieved and a test system is usually complemented by a set of manually performed checks and actions.

4.5 Test case development

Automated interoperability testing implies the availability of abstract and executable test cases specify interactions with one or more EUTs via at least one interface. This clause covers basic guidelines that should be followed when developing software for automated interoperability tests. Not all of them are necessarily specific to the development of interoperability test specifications but are listed here for completeness.

4.5.1 Requirements on test description content

A pre-requisite for the specification of executable test cases is the availability of unambiguous test descriptions. Such descriptions should capture informally all equipment required for a test, pre-conditions, equipment operation and observation, as well as optionally protocol messages (and their contents) or procedures to be checked between EUTs.

Following the existing ETSI testing methodology [3], the result of the interoperability test development process is a collection of interoperability test descriptions (TDs). A TD details test steps that should be followed in order to achieve the test purpose (TP) of each interoperability test. TDs are usually written in natural language and independent of a specific product implementation as well as proprietary interfaces. For automation of interoperability testing, a test case (TC) has to be derived from each test description and should be written in a testing language (e.g., Testing and Test Control Notation (TTCN-3) [6]) or in a scripting language.

Whereas [3] allows the derivation of test case specifications directly from test purposes, this methodology for automated interoperability test development strongly recommends to specify test descriptions as an intermediate step. Test descriptions provide a valuable and easily understandable documentation. Additional test documentation is especially important in interoperability testing due to usually large amount of different interfaces involved in a test as well as its basis on multiple EUTs.

In order to facilitate the specification of test cases an interoperability test description should include as a minimum:

- a unique ID;
- a concise summary of the interoperability test, i.e., its purpose; this description should enable readers to easily and clearly distinguish different tests from each other;
- reference(s) to the base specification section(s) which define the functionality being tested;
- the applicability of a test (in relation with the IFS): recommended, optional, conditional;
- a list of all required equipment for testing and/or test architecture (i.e., their connections);
- test specific pre-conditions that need to be met by the SUT (including configuration required for each equipment);
- the ordered list of equipment operation, i.e., an interoperability test sequence;

- in case of including communication assessment, information about the messages exchanged between equipment under test; here, level of detail of this information depends on desired strength of conformance assessment – one extreme is to assess message field settings as specified in the standard.

4.5.2 Structure, style, and documentation of test cases

As for any software development project also interoperability test suites should be specified using an agreed set of naming conventions that allow the identification of different testing constructs, enforce a proper use of modularization of test software [3], and be properly documented, e.g., using documentation tags [8]. In addition, it is recommended that interoperability test suite specification is based on test libraries [9]. Libraries can help to isolate common interoperability testing constructs, interface specific code, etc and simplify this way the specification of test cases. This also can significantly increase the readability, reusability, and maintainability of test software – especially across several test projects. An example application of these concepts can be found in [10]. Annex A provides an example specification of TTCN-3 interoperability testing library.

In interoperability test cae design, it is recommended to handle each logical interface as well as operation of equipment in with a dedicated of execution thread. In addition, it is recommended to use one execution thread to coordinate and synchronize test case execution. Depending on the test complexity a number of synchronization points should be defined that synchronize either all or only a subset of threads with the test coordination thread.

In order to ensure proper handling of abnormal test termination, the following issues have to be considered interoperability test case design:

- a test case should release test system resources (e.g., interfaces, users, memory) prior to its termination. It should use a general purpose function to manage unpredictable events and stop a test case execution within a specified time limit in case the test system receives no input during a test;
- a test case should attempt to return the SUT to its initial configuration if possible before executing the next test case to minimize the impact of the abnormal test termination on following test case executions.

4.5.3 Test reporting

The test outcome of an interoperability test is usually a complex combination of information from multiple interfaces. An important aspect in automating interoperability testing is that test execution results are properly reported to parties participating in a test or other parties such as the organizers of an interoperability testing event or independent evaluators of a test execution trace. A test report should provide more information than intermediate or final verdicts for each test.

In the context of standards validation the quality of this information is important in order to provide proper feedback to the specification. During interoperability events participants are generally eager to show that it was not their equipment that has caused the error which led to non-interoperability. Detailed test reports help to detect such errors.

NOTE: An error is not the cause of a failure but it leads to a failure. Finding the cause of a failure, i.e., the fault, is a non-trivial task that usually requires a detailed analysis of a test report.

A test report should include the following for each test especially for each failed or inconclusive test:

- the test step in the test description where the test case has failed;
- a description of the difference between the observed and the expected behaviour, i.e., the error or mismatch that leads to the test case failure;
- a reference to the section in a specification which contains the requirement that has been violated;
- the location or interface where the error was observed;
- the complete observation, e.g., received and expected messages.

EXAMPLE: In test TD_IMS_0015, test step 7, call flow message 40A the INVITE message sent by IMS Core Network to UE B the topmost Route header indicating the SIP URI of AS B and containing a Route header indicating the S-CSCF SIP URI of IUT is missing. The received and the expected messages are …

In order to be able to generate such detailed reports, two different approaches can be followed in the test system development process: The first approach leaves the responsibility to generate and attach reports to the verdicts to the test case developer. In the second approach, a report generator is implemented separately from the test system that generates reports by post-processing test execution logs from all interfaces.

The advantage of the first approach is that it provides more flexibility to find the location and origin of an error since the test case developer has a direct control of the code and the ability to transfer directly information into a report from any point in the code. In the second approach, an information loss may arise during the transformation of the reporting related information into a log format which becomes an intermediate format between the testing system and the log analyzer. However, an external log analyzer may have the ability to extract valuable relations between different parts of the logs. This is not possible at design time since all information and logs are built during post-processing analysis.

4.6 Controllability of equipment

Interoperability testing usually requires in practice the operation of one or more equipment, e.g, an EUT or even test equipment, via non standardized interfaces. In a manual test execution, test operators operate this equipment in order to produce the behaviour required by the test. Such information is usually collected using a PICS–like pro-forma in which vendors are asked to specify the operation of their equipment to achieve all behaviour required by the tests.

The following scenarios need to be taken in consideration when trying to automate the control of equipment via non standardized interfaces:

- One or more equipment may need to be configured prior to an interoperability test execution. In some cases this configuration can be handled by the equipment operator once before executing the entire interoperability test suite or multiple tests. But other cases may however require a modification of equipment configuration(s) prior or possibly even also during an interoperability test case execution.
- Frequently one or more equipment may need to be triggered, i.e., stimulated and observed, during a test case execution, e.g., in order to initiate a call from a phone, check that a phone is ringing, etc.
- As part of test configuration some test equipment, e.g., monitoring equipment, may need to be configured, e.g., to configure message filters on specific interfaces for a specific test.

5 Automated interoperability testing of distributed systems

5.1 A general test architecture for interoperability testing

Figure 3 depicts a general architecture for automated interoperability testing. In the following clauses, this architecture and its entities are described in more detail.

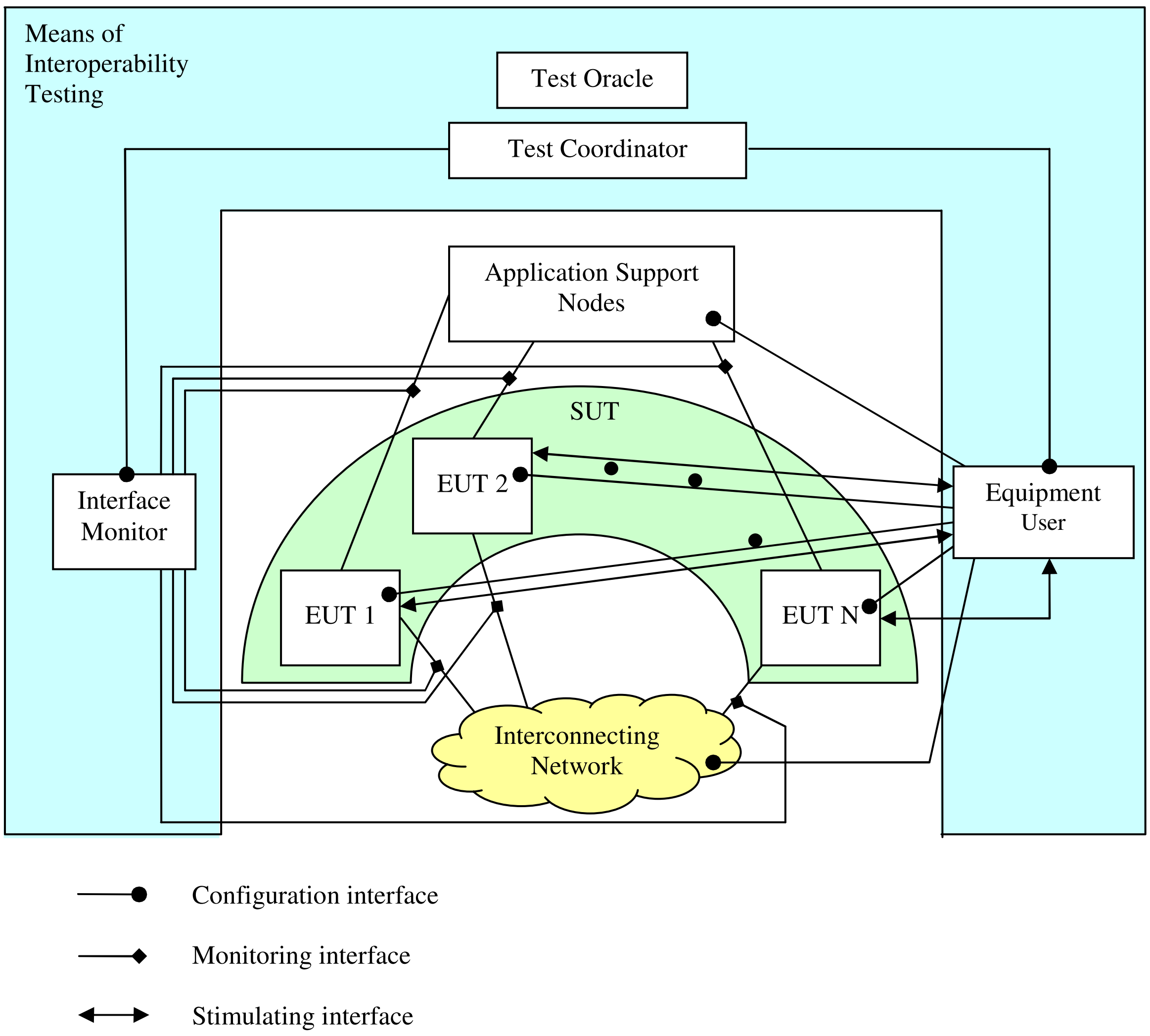

Figure 3: General architecture for interoperability testing

5.1.1 System under test

In interoperability testing, two or more Equipment Under Test (EUT) which constitute the System Under Test (SUT) are assessed for their interoperability. Each EUT is a group of one or more devices usually provided by the same vendor. The concrete device(s) included in a EUT depend on the supplier implementation as well as the service or functionality needed.

5.1.2 Interconnecting network and application support nodes

Interconnecting network and application support nodes are devices essential for the interoperation of the tested equipment and are usually part of the service provided to the end user. They are, however, considered to be neither part of the SUT nor the means of testing. It is assumed that these devices have been previously tested and properly configured with an initial configuration. Their configuration can be changed during test execution.

Interconnecting network includes all the devices needed to provide the required network connections. At least it may be a simple wire. Application support nodes include all the devices involved in providing the service or functionality to the end user which are not object of the test, e.g., a data base with user data.

5.1.3 Means of interoperability testing

The means of interoperability testing includes logical entities that stimulate or control the SUT or configure the interconnecting network as well as the application support nodes. Different types of entities are:

- a equipment user entity that either acts like the end user of the service/functionality provided by an EUT or other equipment or configures remotely an EUT or other equipment like
- the interconnecting network or application support nodes;
- an interface monitor entity that listens and checks to selected EUT interfaces, the interconnecting network or the application support nodes;
- a test coordinator entity that coordinates the execution of all other entities of the means of testing which are assumed to be independent and parallel execution threads. It is in charge of controlling the overall execution, management of testing phases, and synchronization.
- a test oracle entity that collects information manages E2E and conformance verdicts separately.

Depending on existing limitations as well as desired degree of automation a specific instance of a means of interoperability testing may include only a subset of these types of entities as well as multiple instances of each type of entity. In addition, some tasks handled by different entities may be combined by a single execution thread, e.g., test coordinator and oracle.

5.2 A development process for an interoperability test system

Development of a specific interoperability test system is based on specific set of test descriptions, the generic test architecture presented in the previous clause, and limitations applying to the development of a specific test system. A typical development process is illustrated in Figure 4 of which each step is described in more detail in the following clauses. This shown process extends the activity "Write Test Cases" in the process for the general development of interoperability test specifications presented in [3].

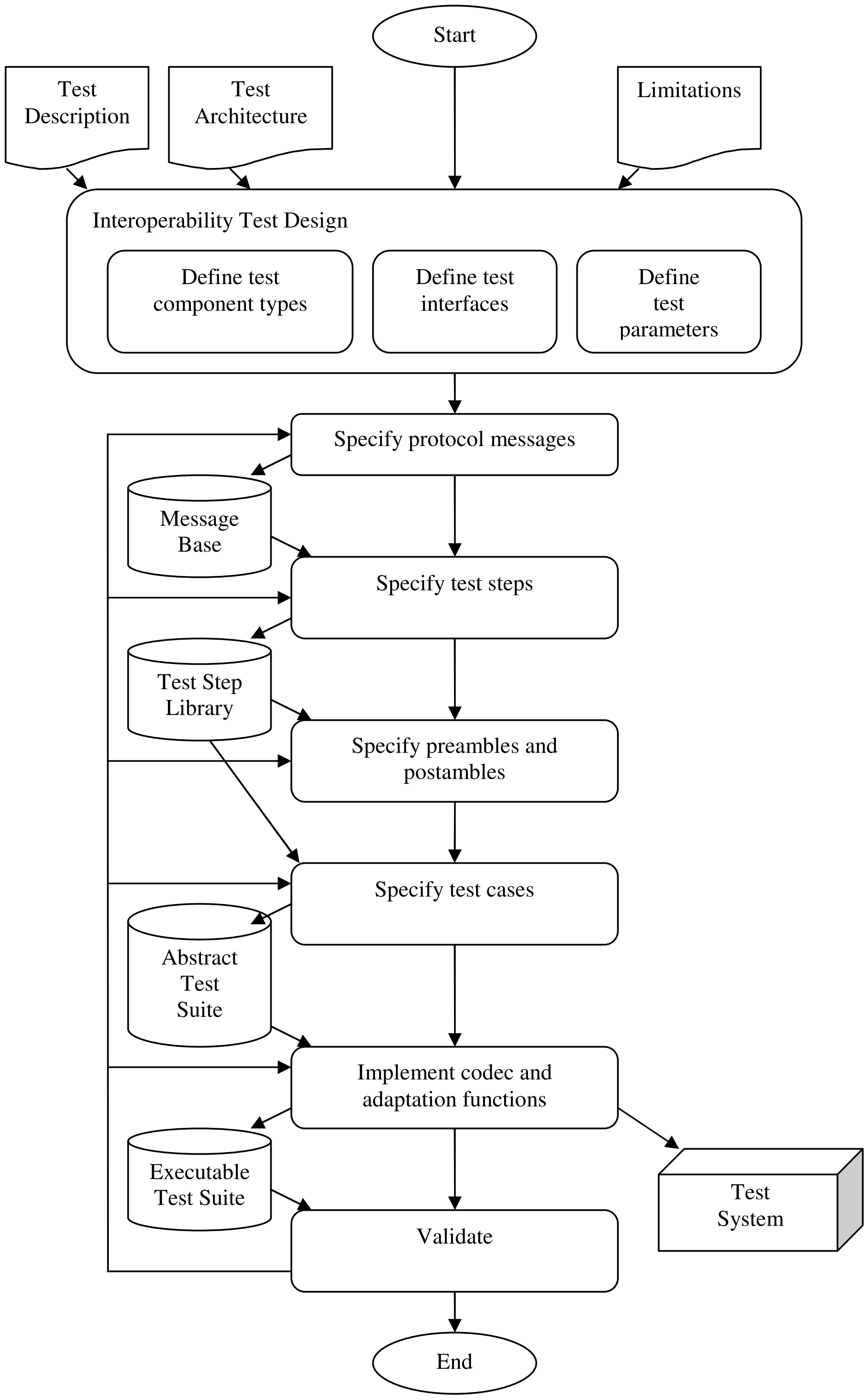

Figure 4: The development process for an interoperability test system

]]>

5.2.1 Design of interoperability tests

5.2.1.1 Definition of test component types

The first step in the design of a test case is the definition of test component types. Each test component type should realize the functionality of one or more types of the test entities a described in clause 5.1.3 and at the same time reproduce the behaviour specified in the test description. Test components should be used to operate specific equipment, monitor relevant EUT interfaces, or configure the communication network. It is recommended to dedicate one test component to handle to operation of one equipment or monitor one specific EUT interface or communication link in order to increase reuse of test behaviour across a test suite and make test case specification less dependent on its availability. Annex B describes two different approaches for the definition of interface monitor test components.

A separate main test component (MTC) should be used as the test coordinator and oracle, i.e., the create and supervise the execution of all other test components running in a test case, to synchronize them on identified synchronization points, and to resolve E2E and conformance interoperability verdicts.

When defining the test components, the designer should pay attention to the logical interface topology of the technology to be tested as well as its configuration. For example, the logical Mw interface of an IMS core network usually maps to multiple IP connections in IMS core network implementations. Similarly, a link set in a SS7 network is distributed on multiple links which can even be distributed on different physical carriers.

5.2.1.2 Define test interfaces

Test interfaces are used by test components to operate equipment and monitor protocol transactions involved in a test. Limitations of automation may prevent the automated access to some interfaces. Some interfaces may not be available at all during test execution.

Proprietary interfaces should be defined as an abstract set of simple commands with parameters that can be mapped to different proprietary interfaces. An example for such a command could be “Initiate a VoIP call” with a parameter “user identifier”. Command definitions should enable a unique association with a specific equipment operation but at the same time be independent of a specific realization. For example, the previous command should not be defined as “Press the green button on the phone” which would restrict the action of making a call to having a keypad available on an EUT.

5.2.1.3 Define test parameters

The ability to parameterize a test case specification is a basic requirement to ensure its adaptability without modification of executable code. Parameterization of test cases needs to be addressed at several levels:

- **EUT information parameters** allow to update test cases EUT information like identifiers, addresses, and port numbers quickly for equipment from different vendors;
- **message parameters** allow to adapt test cases quickly to specific needs of a testing environment or EUT, e.g., use of specific user identities, a specific identifier to be use as a non-existing user, etc;
- **timing parameters** allow to modify timeouts for no activity or other timing like call duration in test cases quickly;
- **interface availability parameters** allow to adapt test cases quickly to the lack of automatic accessibility of certain EUT interfaces prior to the execution of a test
- **test session pairing parameters** allow in the case if interoperability events to quickly select which vendor configuration information is associated with which EUT in the tests,
- **test execution parameters** allow a customized selection of test cases to be executed as part of a test suite, e.g., to not execute tests which are not applicable due to the lack of support of a given functionality by one of the EUTs. They can also be used to run a test suite in a different modes or configurations, e.g., a real-time versus an offline mode.

5.2.2 Specify protocol messages

The second step consists of the definition of test system messages based on the data structures specified for identified test interfaces. This step leads to the definition of the message base. This step might not be necessary if the test specification is written in a formal language (e.g., TTCN-3).

5.2.3 Specify test steps

Each test case can be defined as a sequence of simple and common actions called test steps. Similarly to subroutines and functions in a programming language, it is useful to collect these test steps deriving a test step library. For each test step, a set of parameters will be defined.

The specification of test step behaviour involves:

- contents of messages to be sent and to expect ,
- sequence of actions,
- guarding against deadlock or no activity with timeout,
- handling of exceptions and other unexpected test events,
- setting of conformance and E2E verdicts (partial or final).

As a matter of fact a state machine needs to be defined for each test step. The detail of this state machine specification, in particular for the management of the exception and faults, depends on the accuracy decided according to the testing objectives and constraints.

Which actions and checks to combine in a test step sequence is mainly determined from testing experience. Test descriptions however a good source for the identification of such test steps since they also essentially specify sequences of abstract actions and checks. Therefore, a first step to identify the test steps should be to collect actions and checks from TDs. The association with and reference of test descriptions (or test purposes) in test step specifications hasalso other benefits:

- natural documentation the test case specification;
- easier maintenance of test steps in case of test description changes;
- improved readability of interoperability test execution traces.

5.2.4 Specify preambles and postambles

Each interoperability test requires some initial conditions to be fulfilled; similarly each test requires that the SUT, i.e., all EUTs, is returned to its initial state for the next test to start. The test steps needed to achieve the test preconditions are called the preamble where as all the test steps needed to restore the initial SUT state are called the postambles. Although preamble and postamble do not play a significant role interoperability testing objective itself and no verdicts are associated with them, they are nevertheless useful to separate sequences from the main test behaviour.

5.2.5 Specify test cases

Test cases finally uses all the previous definitions to compose complete test sequences as required by each test description including also pre- and postambles to the test-. Test cases also coordinate independent execution threads part of a test case and compute partial as well as final E2E as well as conformance interoperability verdicts.

This step completes the Abstract Test Suite (ATS) test suite specification. It is written in a formal manner but still not executable as such. The final step need to arrive at an executable test system is the implementation of codec and adaptation functions.

5.2.6 Implement codec and adaptation function

To complete the test system, an implementation of codec and adaptation functions is required. These functions should provide various services necessary for interoperability test execution:

- convert of messages exchanged with the SUT between transfer and abstract protocol syntax and vice versa;
- transport of encoded messages send from test components to correct EUT in the SUT and vice versa;
- handle the connection and conversion of information between the test cases and any other remote/external test equipment, e.g., protocol analyzers, jamming equipment, etc;
- filtering of messages from monitored EUT interfaces
- map abstract EUT operation interfaces to concrete, proprietary interface protocol syntax, e.g., for triggering and configuring EUTs
- trigger and control test case execution;
- present meaningful test execution results to the test operator and generate test reports.

Through compilation and linking the ATS is put together with these functions and an executable code Executable Test Suite (ETS) is created which can be executed against the SUT.

5.2.7 Validation

During the validation step, the test system is connected to a SUT. If a SUT is not available, the test system can be tested in a back to back configuration or with a SUT emulator. The purpose is to assure that the behaviour of test system and its interfaces is compliant to the test description.

Validation realizes a feedback loop to the previous steps in order to correct the message base, the test steps library, the test cases (ATS) or adaptation functions.

After the test system has been checked to be compliant with all TDs, the development is complete and the test system together with the additional set of manual checks and actions introduced which cannot be automated can be used in interoperability testing as described in the ETSI Guide 202 237 [1].

Annex A: TTCN-3 library for interoperability testing

This electronic annex contains as an attachment an example implementation of a TTCN-3 library containing generic constructs matching the concepts introduced in this document and useful for interoperability testing.

TODO include code

Annex B: interface vs. entity approach for realizing the interoperability testing architecture

So far two different approaches have been identified for using interface monitor entities within the general test architecture for interoperability testing as introduced in clause 5.1:

- an interface approach where each interface monitor test component is directly associated with an interface or connection between two EUTs; this approach more straight forward to comprehend from the architecture point of view and more flexible when it comes to dealing with (unexpected lack of) EUT interface availability.
- an entity approach where the monitored interfaces are grouped into higher level testing entities mirroring all EUTs, i.e., the complete architecture of the SUT. This approach allows mirroring message exchanges between EUTs as part of the test execution trace, i.e., as part of the test execution the message exchange specified in test descriptions is produced.

TODO: Illustration

Annex <y>: Bibliography

The annex entitled "Bibliography" is optional.

It shall contain a list of standards, books, articles, or other sources on a particular subject which are not mentioned in the document itself (see clause 12.2 of the EDRs http://portal.etsi.org/edithelp/other/EDRs_Navigator.chm).

It shall not include the following:

1. normative references (such references shall be listed in clause 2.1);
2. informative references (such references shall be listed in clause 2.2).

Use the **Heading 9 style** for the title and B1+ or Normal for the text.

3. <Publication>: "<Title>".

OR

<Publication>: "<Title>".

<PAGE BREAK>

History

Document history		
V0.9.0	Feb 2009	First draft