ETSI TS 102 237-1 V4.1.1 (2003-12)

Technical Specification

Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON) Release 4; Interoperability test methods and approaches; Part 1: Generic approach to interoperability testing

Reference

DTS/TIPHON-06025-1R4

Keywords

H.323, interoperability, IP, SIP, telephony, testing, VoIP

ETSI

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

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

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

Contents

Intellectual Property Rights 6
Foreword 6
1 Scope 7
2 References 7
3 Definitions and abbreviations 8
3.1 Definitions 8
3.2 Abbreviations 8
4 Types of testing 9
4.1 Interoperability testing 9
4.2 Conformance testing 10
4.3 Combining interoperability testing and conformance testing 10
5 Interoperability testing process overview 12
6 Basic concepts 12
6.1 Means of Testing 13
6.2 Equipment Under Test (EUT) 13
6.3 Qualified Equipment (QE) 13
6.3.1 QEs and Devices 13
6.3.2 Designating the first QE 14
6.4 System Under Test (SUT) 14
6.5 Test interface 14
6.6 Test driver 14
6.7 Test coordinator 15
6.8 Interoperability test cases 15
6.9 Means of Communication 15
7 Generic interoperability test architectures 15
7.1 Test architectures with a single QE 16
7.2 Test architectures with multiple QEs 16
7.2.1 An example using 3 QEs 17
7.2.2 Testing IP hosts with multiple QEs 17
8 Developing interoperability tests 18
8.1 Overview 18
8.2 Specify abstract architecture 19
8.3 Prepare draft IFS Proforma 19
8.4 Specify Test Suite Structure 20
8.4.1 Identify test groups 20
8.4.2 Define test coverage within each test group 20
8.5 Write Test Purposes 21
8.6 Write test cases 21
8.6.1 Pre-test conditions 21
8.6.2 Test steps and verdicts 21
8.6.2.1 Test steps 21
8.6.2.2 Verdicts 21
8.6.2.3 Specification of test steps and verdicts 22
8.6.3 Example 22
8.6.4 Pre-amble and post-amble 23
8.6.4.1 Alternative test case presentation forms 23
8.7 Validate test cases 28
8.8 Finalize IFS 28
9 Interoperability testing process 28
9.1 Overview 28
9.2 Prepare for testing 29

9.2.1 Test arrangement 29
9.2.2 Test planning 30
9.3 Testing....30
9.3.1 Manual testing 30
9.3.2 Automated testing 31
9.4 Write test report....31

Annex A (informative): Example IFS (TIPHON Profile of H.323, Release 3)....32

A.1 Introduction 32

A.2 Instructions for completing the IFS proforma....32
A.2.1 General structure of the IFS proforma....32
A.2.2 Additional information 32

A.3 IFS proforma 33
A.3.1 Implementation identification 33
A.3.2 Protocol summary....33

A.4 H.323 entities 33
A.4.1 Roles....33
A.4.2 Terminal capabilities 34
A.4.2.1 Gatekeeper discovery....34
A.4.2.2 Registration....34
A.4.2.3 Basic call....35
A.4.2.4 Supplementary services 35
A.4.3 Gatekeeper capabilities....35
A.4.3.1 Gatekeeper in the serving network 35
A.4.3.1.1 Overall capabilities....35
A.4.3.1.2 Gatekeeper discovery....35
A.4.3.1.3 Registration 36
A.4.3.1.4 Basic call....36
A.4.3.1.5 Supplementary services....36
A.4.3.2 Gatekeeper in the intermediate network 36
A.4.3.2.1 Overall capabilities....36
A.4.3.2.2 Gatekeeper discovery....36
A.4.3.2.3 Registration 37
A.4.3.2.4 Basic call....37
A.4.3.2.5 Supplementary services....37
A.4.3.3 Gatekeeper in the home network 37
A.4.3.3.1 Overall capabilities....37
A.4.3.3.2 Gatekeeper discovery....37
A.4.3.3.3 Registration 38
A.4.3.3.4 Basic call....38
A.4.3.3.5 Supplementary services....38
A.4.4 Gateway capabilities....38
A.4.4.1 Overall capabilities 38
A.4.4.2 Basic call....39
A.4.4.3 Supplementary services 39

Annex B (informative): Example IFS (TIPHON Profile of SIP, Release 3) 40

B.1 Introduction 40

B.2 Instructions for completing the IFS proforma....40
B.2.1 General structure of the IFS proforma....40
B.2.2 Additional information 40

B.3 IFS proforma 41
B.3.1 Implementation identification 41
B.3.2 Protocol Summary, EN 301 xxx....41

B.4 SIP entities....41
B.4.1 Roles....41
B.4.2 User Agent capabilities 42

B.4.2.1 Registration....42
B.4.2.2 Basic call....42
B.4.3 Registrar capabilities....43
B.4.3.1 Registration....43
B.4.4 Proxy capabilities....43
B.4.4.1 Proxy in the serving and intermediate network....43
B.4.4.1.1 Registration....43
B.4.4.1.2 Basic call....43
B.4.4.2 Proxy in the home network....44
B.4.4.2.1 Registration....44
B.4.4.2.2 Basic call....44
B.4.5 Gateway capabilities....44
B.4.5.1 Basic call....44

Annex C (informative): Experimental testbed....45

C.1 Description of an experimental TIPHON H.323 - TIPHON SIP interoperability testbed....45
C.1.1 TIPHON H.323 - TIPHON SIP interoperability testing documentation....45
C.1.2 IP infrastructure....46
C.1.3 Requirements for VoIP applications....47
C.1.4 Monitoring facilities....48

C.2 The experimental H.323-SIP testbed....48
C.2.1 Components of the interoperability testbed....48
C.2.1.1 Supporting companies....48
C.2.1.2 Open source software....49
C.2.2 Interoperability testing activities....49
C.2.3 Interoperability test specification documentation....49
C.2.4 Abstract architecture....50
C.2.5 Testing arrangements....50
C.2.5.1 BIT....50
C.2.5.2 H.323 interoperability testing and SIP interoperability testing....51
C.2.5.3 H.323-SIP interoperability testing....52
C.2.6 Logging....53
C.2.7 Example result matrices....53
C.2.8 Reporting....54

Annex D (informative): Bibliography....55

History....56

Intellectual Property Rights

Foreword

This Technical Specification (TS) has been produced by ETSI Project Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON).

The present document is part 1 of a multi-part deliverable covering Interoperability test methods and approaches, as identified below:

Part 1: "Generic approach to interoperability testing";

Part 2: "H.323-SIP interoperability test scenarios to support multimedia communications in NGN environments".

1 Scope

The present document, "A generic approach to interoperability testing", gives general guidance on the specification and execution of interoperability tests for communication systems in Next Generation Networks (NGN). It provides a framework within which interoperability test specifications for a wide range of product types can be developed. The guidelines are expressed as recommendations rather than strict rules and leave enough freedom to allow test specifiers to adopt and adapt processes to suit each particular project while still ensuring that test specifications accurately reflect the requirements of the base standards and can be executed consistently across a range of configurations.

Interoperability testing is the structured and formal testing of functions supported remotely by two or more items of equipment communicating by means of standardized protocols. It is not the detailed verification of protocol requirements specified in a conformance test suite, neither is it the less formal development testing often associated with plug-fest and interop events (frequently referred to as "bake-offs").

Although some consideration is given within the methodology to the operating and reporting aspects of interoperability testing, the primary focus of the present document is on the specification of interoperability testing architectures, test plans and test suites.

2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication and/or edition number or version number) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies.

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

[1] ISO/IEC 9646 (parts 1 to 7): "Information technology - Open Systems Interconnection - Conformance testing methodology and framework".

[2] IETF RFC 3261: "SIP: Session Initiation Protocol".

[3] ETSI EG 202 107: "Methods for Testing and Specification (MTS); Planning for validation and testing in the standards-making process".

[4] ETSI TS 102 237-2: "Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON) Release 4; Interoperability Test Scenarios; Part 2: H.323-SIP interoperability test scenarios to support multimedia communications in NGN environments".

[5] ITU-T Recommendation H.323: "Packet-based multimedia communications systems".

[6] ETSI TS 101 883: "Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON) Release 3; Technology Mapping; Implementation of TIPHON architecture using H.323".

[7] ETSI TS 101 884: "Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON) Release 3; Technology Mapping; Implementation of TIPHON architecture using SIP".

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the following terms and definitions apply:

conformance: compliance with requirements specified in applicable standards ISO/IEC 9646 [1]

conformance testing: testing the extent to which an Implementation Under Test (IUT) satisfies both static and dynamic conformance requirements ISO/IEC 9646 [1]

NOTE The purpose of conformance testing is to determine to what extent a single implementation of a particular standard conforms to the individual requirements of that standard.

device: item of software or hardware which either alone or in combination with other devices implements the requirements of a standardized specification

Equipment Under Test (EUT): grouping of one or more devices which has not been previously shown to interoperate with previously Qualified Equipment (QE)

interoperability: ability of two systems to interoperate using the same communication protocol

interoperability testing: activity of proving that end-to-end functionality between (at least) two communicating systems is as required by the base standard(s) on which those systems are based

interoperability test suite: collection of test cases designed to prove the ability of two (or more) systems to interoperate

InterWorking Function (IWF): translation of one protocol into another one so that two systems using two different communication protocols are able to interoperate

Qualified Equipment (QE): grouping of one or more devices that has been shown, by rigorous and well-defined testing, to interoperate with other equipment

NOTE: Once an EUT has been successfully tested against a QE, it may be considered to be a QE, itself.

System Under Test (SUT): one or more QEs and an EUT

3.2 Abbreviations

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

API	Application Programming Interface
BIT	Basic Interconnection Tests
EP	End Point
EUT	Equipment Under Test
GFT	Graphical presentation Format for TTCN-3
GK	GateKeeper
ICS	Implementation Conformance Statement
IFS	Interoperable Features Statement
IUT	Implementation Under Test
IWF	InterWorking Function
MMI	Man-Machine Interface
MoC	Means of Communication
MoT	Means of Testing
NDA	Non-Disclosure Agreement
NE	Network Element
NGN	Next Generation Network
PCO	Point of Control and Observation
PICS	Protocol Implementation Conformance Statement
QE	Qualified Equipment

SIP	Session Initiation Protocol
SUT	System Under Test
TE	Terminal Element
TP	Test Purpose
TSS	Test Suite Structure

4 Types of testing

Conformance testing establishes whether or not the implementation in question meets all of the requirements specified for the protocol itself. For example, it will test protocol message contents and format as well as the permitted sequences of messages.

Equipment implementing standardized protocols and services can be formally tested in two related but different ways.

Individually, each of these test approaches has benefits and limitations. Conformance testing can show that a product correctly implements a particular standardized protocol while interoperability testing can demonstrate that it will work with other like products. Conformance testing prior to interoperability testing provides both the proof of conformance and the guarantee of interoperation.

Interoperability testing assesses the ability of the implementation to support the required trans-network functionality between itself and another, similar implementation to which it is connected.

4.1 Interoperability testing

The term "interoperability testing" is often used in relation to the semi-formal testing carried out at multi-vendor events as part of the product development process. While such events, often referred to as "plug-fest", "interops" and "bake-offs", are valuable sources of information on the ability of a product to communicate, they do not offer the structured, and, therefore, repeatable, testing that is an essential part of a certification scheme. For a certification (or branding or logo) scheme to be meaningful, it is necessary that interoperability testing is carried out in accordance with a comprehensive and structured suite of tests. In the context of the present document, it is exactly this type of testing which is referred to as "interoperability testing". For other types of schemes, such as those arranged between manufacturers for marketing or other purposes this approach is still valid.

NOTE: It is possible that other organizations within the global standardization community will have interpretations of this term which differ to a greater or lesser extent.

The purpose of interoperability testing is to prove that end-to-end functionality between (at least) two communicating systems is as required by the standard(s) on which those systems are based.

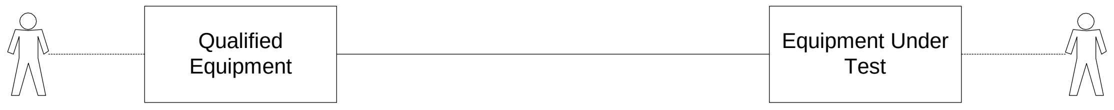


Figure 1: Illustration of interoperability testing

The important factors which characterize interoperability testing are:

- the Equipment Under Test (EUT) and the Qualified Equipment (QE) together define the boundaries for testing (figure 1);
- the EUT and QE come from different suppliers (or, at least, different product lines);
- interoperability tests are performed at interfaces that offer only normal user control and observation;
- interoperability tests are based on functionality as experienced by a user (i.e. they are not specified at the protocol level). In this context a user may be human or a software application;
- the tests are performed and observed at functional interfaces such as Man-Machine Interfaces (MMIs), protocol service interfaces and Application Programming Interfaces (APIs).

The fact that interoperability tests are performed at the end points and at functional interfaces means that interoperability test cases can only specify functional behaviour. They cannot explicitly cause or test protocol error behaviour.

4.2 Conformance testing

The purpose of conformance testing is to determine to what extent a single implementation of a particular standard conforms to the individual requirements of that standard.

Conformance Test System
System Under Test

Figure 2: Illustration of conformance testing

The important factors which characterize conformance testing are as follows:

- the System or Implementation Under Test (SUT or IUT) defines the boundaries for testing (figure 2);
- the tests are executed by a dedicated test system that has full control and observability;
- the tests are performed at open standardized interfaces that are not (usually) accessible to a normal user. (i.e. they are specified at the protocol level).

Because the conformance tester maintains a high degree of control over the sequence and contents of the protocol messages sent to the IUT it is able to be comprehensive in that it can explore a wide range of both expected and unexpected (invalid) behaviour.

It is not within the scope of the present document to define conformance testing methodology. However, because interoperability testing and conformance testing complement one another the reader of the present document would be well-advised to study the established ISO conformance testing methodology defined in ISO/IEC 9646 parts 1 to 7 [1] as applied in all ETSI conformance test specifications.

4.3 Combining interoperability testing and conformance testing

In common with most standardization bodies, ETSI has responsibilities for producing test specifications, both conformance and interoperability, but not for undertaking actual testing. Figure 3 shows how the development of communication standards and of test specifications fall within ETSI's area of responsibility but that conformance testing, interoperability and certification are all outside this area.

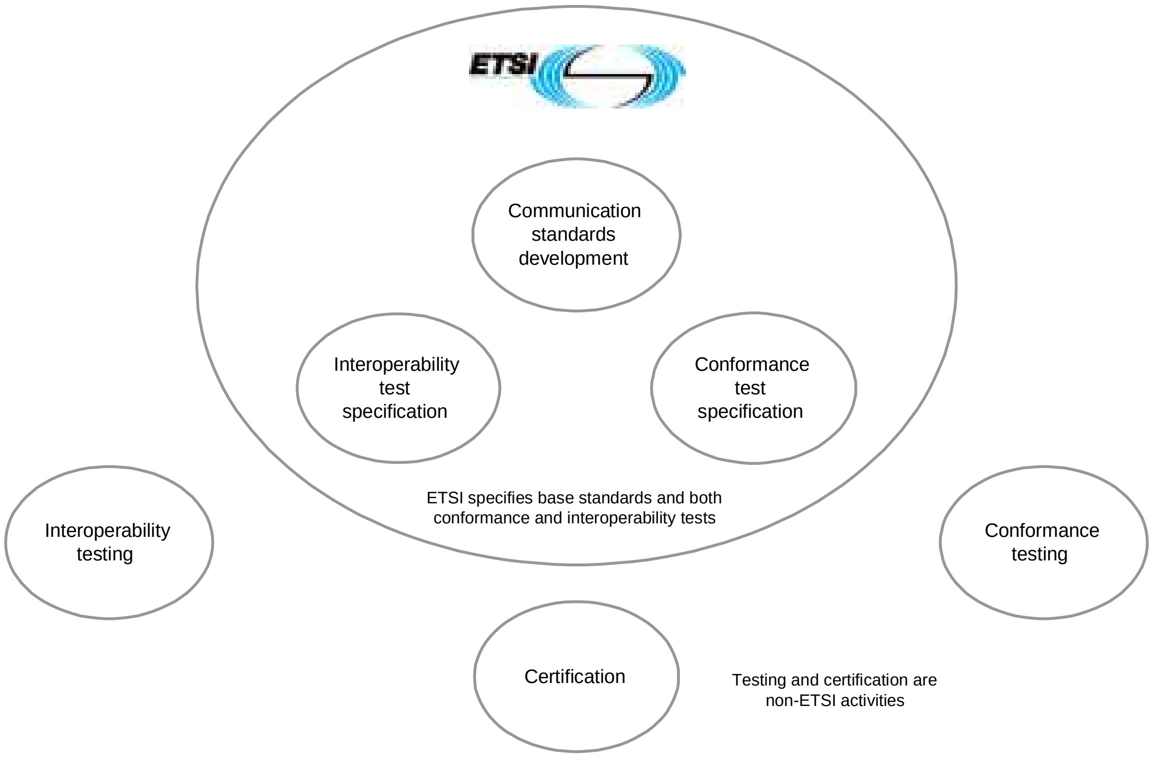

ETSI
Communication standards development
Interoperability test specification
Conformance test specification
ETSI specifies base standards and both conformance and interoperability tests
Interoperability testing
Conformance testing
Certification
Testing and certification are non-ETSI activities

Figure 3: Overall relationship of testing activities

Conformance and interoperability are both important and useful approaches to the testing of standardized protocol implementations although it is unlikely that one will ever fully replace the other. Conformance testing is able to show that a particular implementation complies with all of the protocol requirements specified in the associated base standard. However, it is difficult for such testing to be able to prove that the implementation will interoperate with similar implementations in other products. On the other hand, interoperability testing can clearly demonstrate that two implementations will cooperate to provide the specified end-to-end functions but cannot easily prove that either of them conforms to the detailed requirements of the protocol specification.

The purpose of interoperability testing is not only to show that products from different manufacturers can work together but also to show that these products can interoperate using a specific protocol. Without this additional aspect, interoperability testing could be considered to be almost meaningless. Within the context of standardization, it is of little interest to know that two products can interoperate unless there is a guarantee that they are connected together by means of a standardized protocol. It is, therefore, advisable to conformance test an implementation before testing for interoperability with other (similarly tested) implementations.

Although there are quite distinct differences between conformance testing and interoperability testing, it is valid to consider using the techniques together to give combined results. Such an approach will almost certainly involve some compromise and it is unlikely that it would provide the breadth and depth of testing that conformance and interoperability can offer when applied individually. However, some limited conformance testing with extensive interoperability testing, for example, may be useful in certain situations. The test configuration shown in figure 4 permits complete interoperability testing to be undertaken while limited protocol conformance monitoring takes place.

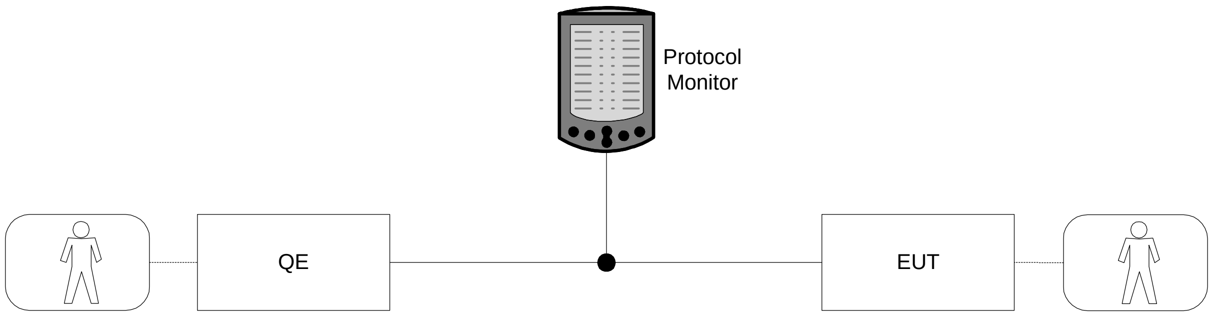


Figure 4: Interoperability testing with conformance monitoring

While this arrangement cannot provide a complete proof of conformance, analysis of the protocol monitor output will be able to show whether protocol signalling between the IUT and QE conformed to the appropriate standard(s) throughout the testing.

5 Interoperability testing process overview

The present document provides users with guidelines on the main steps associated with interoperability testing. The intention is that the guidelines should be simple and pragmatic so that the document can be used as a "cook-book" rather than a rigid prescription of how to perform interoperability testing.

The main components of the guidelines are described in clauses 8 and 9 and are as follows:

- development of interoperability test specifications, including:
  - identification of interoperable functions;
  - identification of abstract architectures;
  - specification of interoperability test suite structure and test purposes;
  - specification of interoperability test cases;
- the testing process, including:
  - test planning;
  - specification of test configurations;
  - execution of the tests;
  - logging results and producing test reports.

As their name implies, guidelines are only for guidance and the actual process followed should use and adapt whichever of these guidelines are most applicable in each particular situation. In some cases this may mean the application of all aspects.

6 Basic concepts

Figure 5 illustrates the main concepts presented in the present document. It shows the two main components of the methodology, namely the Means of Testing (MoT) and the System Under Test (SUT). The MoT includes the roles of test drivers and a test coordinator, the interoperability test cases and mechanisms for logging and reporting. The SUT comprises the Equipment Under Test (EUT) and the Qualified Equipment (QE). The Means of Communication (MoC) between the QE and the EUT is considered to be neither part of the SUT nor of the MoT.

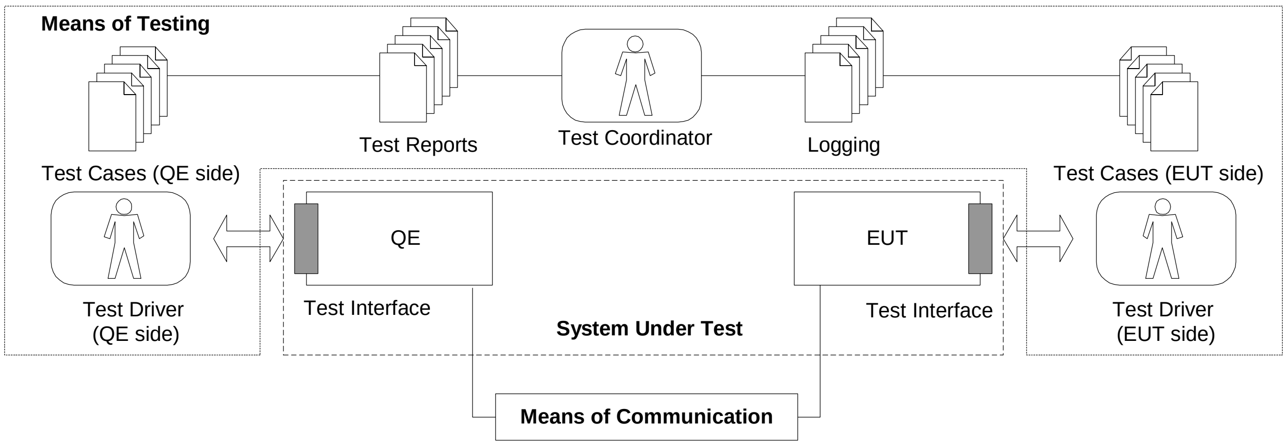


Figure 5: Illustration of main concepts

6.1 Means of Testing

The combination of equipment and procedures that perform the selection and execution of test cases is known as the Means of Testing (MoT). Execution of test cases may be achieved either by a human operator or by an automated program (see clause 6.6). The MoT should also be capable of logging test results and of producing test reports (see clause 9.4). The MoT includes neither the System Under Test nor the means by which devices in the System Under Test communicate.

6.2 Equipment Under Test (EUT)

In any interoperability testing architecture there will always be one connected item which is the subject of the test. This item is referred to as the Equipment Under Test or EUT. Any single test configuration will only have one EUT. An EUT may be end-user equipment (such as a terminal), network equipment (such as a router) or a software application.

EUTs can be composed of any number of component parts each of which is referred to as a device. This may be a physical device, a software package or a combination of the two. The simplest case is where the EUT is a single device. An EUT cannot be decomposed into sub-EUTs.

The interconnection configuration between devices in an EUT is purely a matter for the supplier and is not prescribed in the test architectures, nor is it considered to be an explicit part of the interoperability test for that EUT.

An EUT will not have been previously tested for interoperability in a similar configuration although it may have been tested for conformance. While this methodology does not require previous conformance testing, it is recommended that this activity is performed, for the reasons mentioned in clause 4.3.

6.3 Qualified Equipment (QE)

6.3.1 QEs and Devices

When testing an EUT for interoperability, it is essential that the test architecture includes equipment that has already been proven to interoperate with similar equipment from other suppliers. Such items are referred to as the Qualified Equipment (QE). Any single test configuration may have one or more QEs. A QE may be end-user equipment (such as a terminal), network equipment (such as a router) or a software application.

QEs can also be composed of a number of component parts, each of which is, again, referred to as a device. This may be a physical device, a software package or a combination of the two. The simplest case is where the QE is a single device. A QE cannot be decomposed into sub-QEs. Thus, a QE is a collection of devices that, in a given configuration, have undergone and passed interoperability testing. However, in the context of being "inside" another QE (i.e. in that particular configuration) it is cannot be considered to be acting as a QE. In this methodology, it is left to each testing scheme to define the rules on what constitutes a valid QE and the distinction between a QE and the devices of which it is composed.

The interconnection configuration between devices in a QE is purely a matter for the test system implementer and is not prescribed in the test architectures.

Any given QE will have initially been tested as an EUT but, once the full range of interoperability tests have been successfully performed, it can be considered to be a QE. This methodology does not force an EUT to be tested against all possible QEs in the pool of QEs that may be available in a particular testing scheme. However, the likelihood of multi-vendor interoperability is increased if it can be demonstrated that a particular EUT interoperates with a large number of different QEs.

6.3.2 Designating the first QE

In cases of new and developing technologies, no Qualified Equipment is likely to exist. The first instance of interoperability testing for a particular scheme will involve two (or more) EUTs rather than a number of QEs and one EUT.

Once these EUTs are shown to successfully interoperate, they will all be designated as QEs with none having precedence over any other. The testing scheme can then continue with new EUTs joining the pool of the existing QEs that have already been tested in a given configuration.

It is strongly recommended that both the two initial EUTs have undergone conformance testing prior to interoperability testing.

6.4 System Under Test (SUT)

The System Under Test (SUT) is the combination of one or more QEs and one single EUT.

6.5 Test interface

The interfaces that are made available by the SUT in order to perform testing are known as the test interfaces. These interfaces are accessed by the test driver. Interfaces internal to the SUT may be used for logging and/or analysis but they are not considered to be an essential part of the test configuration.

In the simplest case, a test interface will be the normal user interfaces offered by the product undergoing testing (EUT) and/or by the QEs that are part of the SUT. Terminal equipment, for example, may be tested using a keypad, or a point-and-click dialog, or a combination of the two. Other cases, such as protocol stacks, may offer an API over which automated interoperability testing can be performed.

An SUT will offer at least one interface to either the test driver and/or the QEs.

6.6 Test driver

As interoperability testing involves control and observation at the functional (rather than signalling) level, interoperability tests should be described in terms of activities by the user of the endpoint equipment. In many cases, this user can be considered to be a human but in others it will be more appropriate to think of the user as an application within a software system.

As a means of improving testing efficiency and consistency, the role of the test driver may be performed by an automatic device programmed to carry out the specified test steps.

The following examples illustrate both of these cases:

EXAMPLE 1: *Human User:* A test architecture is established for VoIP interworking with two telephony terminals and two routers connected together using IP. Interoperability tests are specified at the terminals in terms such as "Take telephone A off-hook; Dial the E.164 number of telephone B etc.".

EXAMPLE 2: *Application User:* A test architecture is established for SIP interoperability with two routers connected together but no user terminals because at the time of testing there are no suitable applications available. Interoperability tests are specified in terms such as "Cause INVITE message to be sent from QE to IP address at EUT; On receipt of INVITE from QE, cause 100 TRYING message to be sent from EUT to QE; etc.".

In the first case, the human test driver will be performing valid tasks of a normal user of the system, using only the interfaces (e.g. MMI) offered by a product. In the second case, the test driver will be manipulating the EUT and the QE by whatever means is possible (for example, over an API) to ensure that specific messages are sent and observed.

6.7 Test coordinator

In any given instance of testing there will be at least two interfaces over which the tests will be run (see clause 6.5). The test coordinator is responsible for synchronizing the actions of the two (or more) test drivers, if needed. The test coordinator is only a conceptual role and, in a practical case of testing, this role may be taken by, for instance, one of the test drivers.

6.8 Interoperability test cases

An Interoperability test case is the detailed set of instructions (or steps) that need to be taken in order to perform the test. In the case where the test driver is a human operator, these instructions will be in natural language (see clause 8.6). In the case where the tests are automated, they may be written in a programming or test language such as TTCN-3. The combined test cases should cover all events at each of the available test interfaces.

6.9 Means of Communication

The QE and EUT are connected by the Means of Communication (MoC). This, for example, may be a simple wire or a complex network of interconnected devices. In all cases this underlying transport mechanism is not considered to be part of the SUT.

It is assumed that the underlying communication layers have been tested (i.e. are conformant).

7 Generic interoperability test architectures

Figure 6 shows a generic architecture for interoperability testing. All interoperability testing architectures that show the relationship between the EUT, the QEs and the test operators can be derived from this model. The test driver for the EUT is optional, depending on the kind of equipment being tested. As an example, an EUT which is an interworking function (see clause 7.2) would probably not require a test driver function.

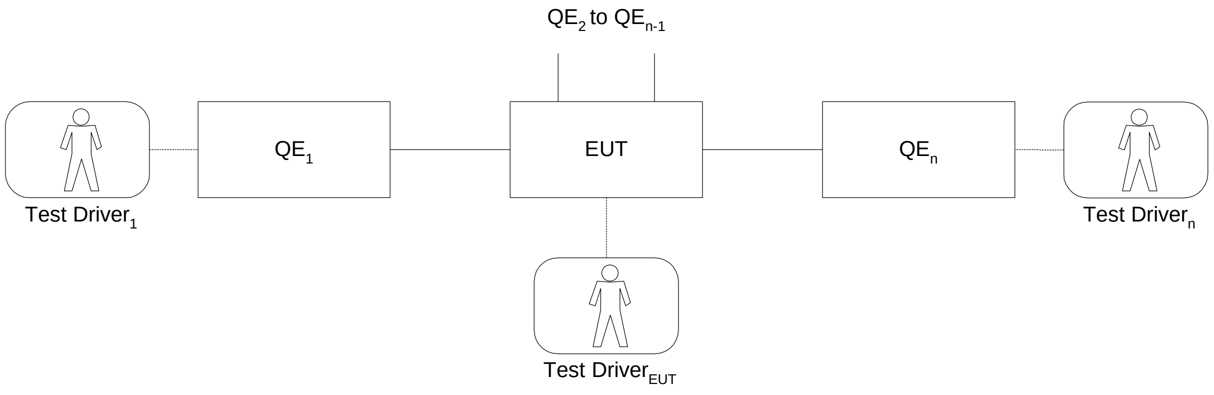


Figure 6: Generalized interoperability testing architecture

For simplicity, this figure shows that the QE and the EUT offer only a single interface to a single test driver. However, it is possible that an EUT or QE could offer more than one interface to one or more test drivers. This relationship need not necessarily be a one-to-one mapping.

7.1 Test architectures with a single QE

Figure 7 shows the simplest architecture where there is only one QE.

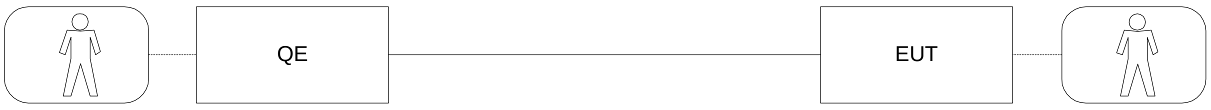


Figure 7: Basic interoperability test architecture

A typical example of this would be the case of testing terminal equipment such as a SIP phone from a given manufacturer. The QE is a SIP phone (from a different manufacturer) that has been tested previously. This is illustrated in figure 8.

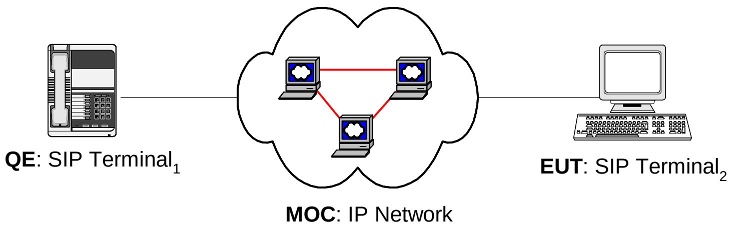


Figure 8: Example of the basic interoperability test architecture for SIP phones

7.2 Test architectures with multiple QEs

Figure 9 shows the generic architecture for n=2 and with no test driver for the EUT.

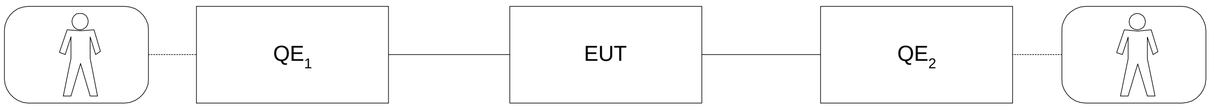


Figure 9: Basic interoperability test architecture with n=2 and no EUT test driver

Modern voice and multi-media communications use a range of protocol signalling systems which, in most cases, are not directly compatible. Equipment within a network that can translate the signalling protocol of one system to that of another is often referred to as an InterWorking Function (IWF). The test architecture of figure 9 is applicable to testing IWFs. The case of SIP-H.323 IWF is illustrated in figure 10.

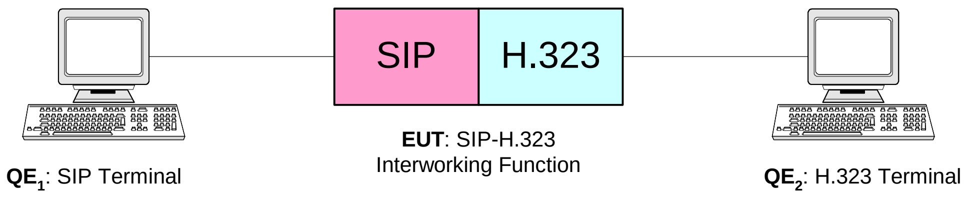

SIP
H.323
QE$_1$: SIP Terminal
EUT: SIP-H.323 Interworking Function
QE$_2$: H.323 Terminal

Figure 10: Interoperability test architecture for SIP⇔H.323 InterWorking Function

7.2.1 An example using 3 QEs

Figure 11 shows the generic architecture with 3 QEs and with no test driver for the EUT.

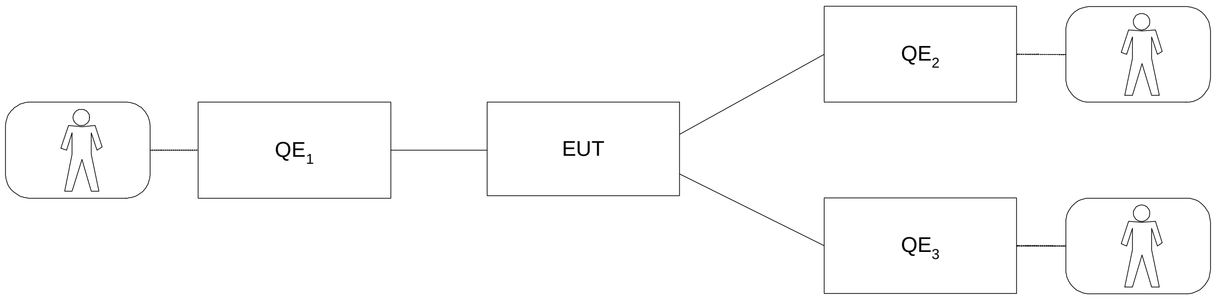


Figure 11: Basic interoperability architecture with 3 QEs and no EUT test driver

A concrete example of this architecture is shown in figure 12 which shows interoperability testing of the call diversion service using three QEs; one to make a call and two to show that the transfer has indeed taken place.

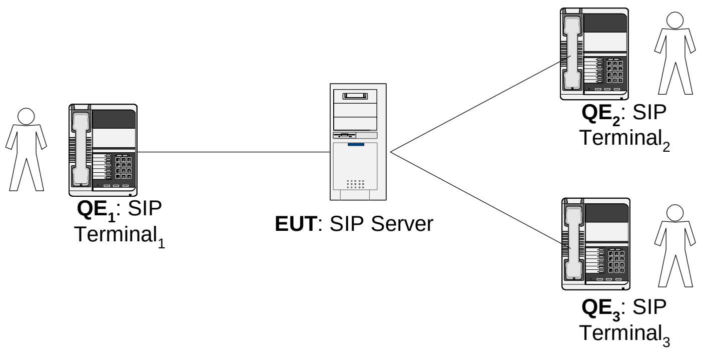

QE$_2$: SIP Terminal$_2$
QE$_1$: SIP Terminal$_1$
EUT: SIP Server
QE$_3$: SIP Terminal$_3$

Figure 12: Using three QEs to test the call diversion service

NOTE: It is possible to draw a box round QE1 and QE2 and illustrate the configuration as having only two QEs. However, for clarity it is far better to show the QEs as two separate pieces of equipment.

7.2.2 Testing IP hosts with multiple QEs

Figure 13 shows a more complex architecture for testing the interoperability of an Internet host (Host$_4$) with routers and other hosts. The Means of Communication in this architecture is the Internet cloud and the Ethernet local network. Because the interplay between the two routers and host is a key part of the test the routers are not included in the MoC.

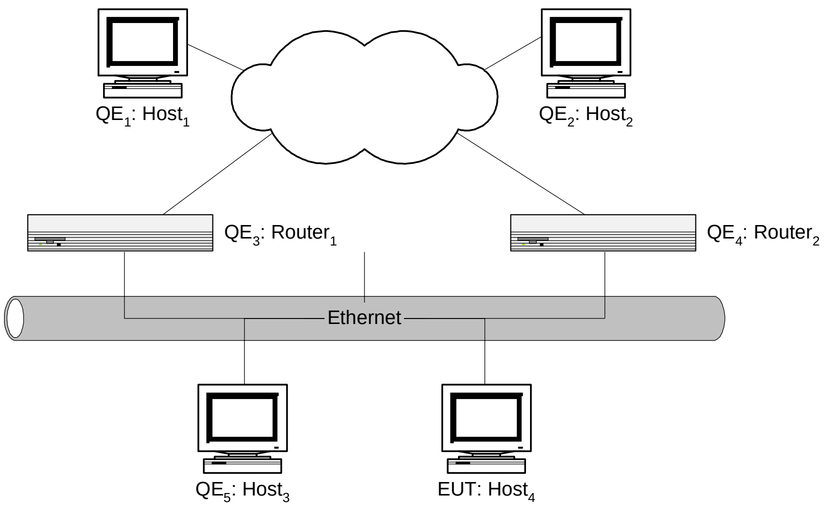


Figure 13: Interoperability testing an IP host with multiple QEs

8 Developing interoperability tests

8.1 Overview

The development of an interoperability test specification should follow a similar path to that taken when developing a conformance test specification. A close parallel can also be seen between the component parts of each type of test specification.

The steps involved in the process of developing an interoperability test specification are as follows:

- specify abstract architecture;
- prepare draft Interoperable Features Statement (IFS);
- specify Test Suite Structure (TSS);
- write Test Purposes (TP);
- write test cases;
- validate test cases;
- finalize IFS.

This process is expressed graphically in figure 14 using a UML activity diagram.

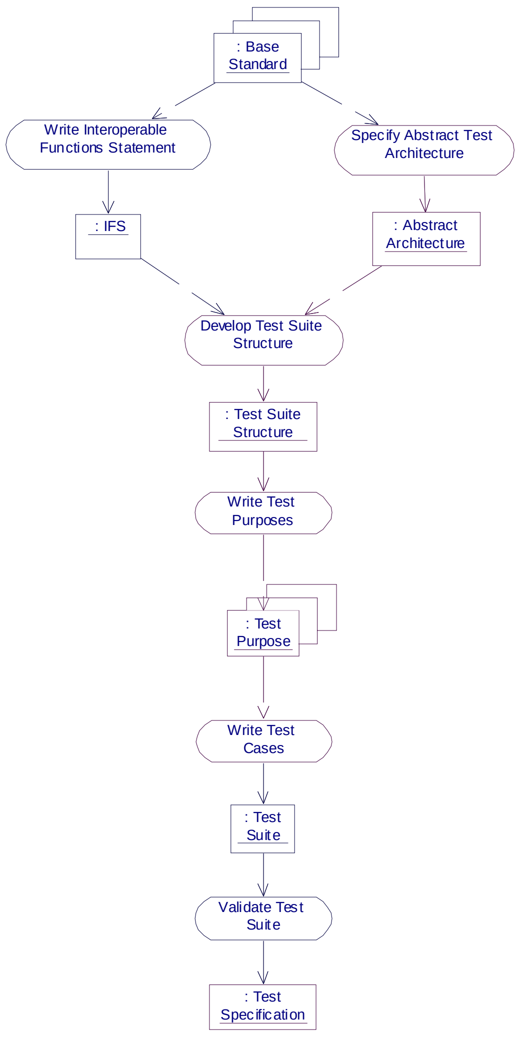


Figure 14: Developing an interoperability test specification

8.2 Specify abstract architecture

An abstract testing architecture provides a general framework within which specific test arrangements must fit in order to perform the specified suite of tests. Defining this architecture at an early stage should help to provide a structure for the test cases specified later. Abstract architectures can be expressed in diagrammatic, tabular or textual form and should clearly identify:

- the EUT;
- the QE(s);
- the communications paths between the EUT and QE(s);
- valid types of equipment for the EUT and QE(s);
- if required, the expected protocol to be used in providing communication between the EUT and QE(s).

Figure 15 shows in diagrammatic form an example of an abstract architecture for the testing of a stateful SIP proxy. In this example, one SIP proxy is identified as the EUT with another proxy plus two SIP end points identified as QEs. The Means of Communication is not specified although it is implied that it must carry SIP.

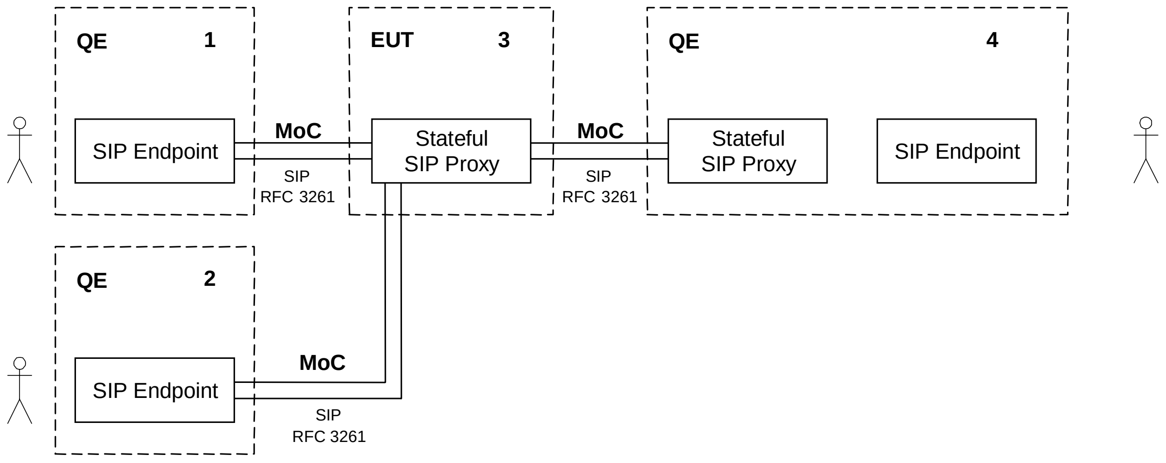


Figure 15: Example abstract architecture diagram

This abstract architecture could equally well be represented in a table, as shown in table 1.

Table 1: Example abstract architecture table

Item	EUT/QE	Equipment type	Connected to item	MoC
1	QE	SIP Endpoint	3	SIP (RFC 3261 [2])
2	QE	SIP Endpoint	3	SIP (RFC 3261 [2])
3	EUT	Stateful SIP Proxy	1	SIP (RFC 3261 [2])
			2	SIP (RFC 3261 [2])
			4	SIP (RFC 3261 [2])
4	QE	Stateful SIP Proxy + SIP Endpoint	3	SIP (RFC 3261 [2])

The abstract architecture should be derived from the requirements of the base protocol standard(s), and should be specified in a form that makes it simple to map each element of a concrete test scenario to it.

8.3 Prepare draft IFS Proforma

The purpose of an Interoperable Features Statement (IFS) is to identify those standardized functions which an EUT must support, those which are optional and those which are conditional on the presence 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.

In addition, the IFS can be used as a proforma by a manufacturer in identifying which functions an EUT will support when interoperating with similar equipment from other manufacturers.

If it exists, the ideal starting point in the development of an IFS is the Protocol Implementation Conformance Statement (PICS) which should clearly identify the options and conditions which apply to the protocol to be tested. Like the PICS, the IFS should be considered part of the base protocol specification and not a testing document.

At this stage of the test suite development, the IFS can only be considered as a complete draft. As the test suite evolves, it is possible that errors and omissions in the IFS will be identified. These should be recorded for correction at a later stage (see clause 8.8). Example IFSs (for the TIPHON profile of both ITU-T Recommendation H.323 [5] and IETF SIP) can be found in annexes A and B.

8.4 Specify Test Suite Structure

8.4.1 Identify test groups

There is no hard and fast rule that can be used to determine how a test suite should be divided up into test groups other than to say that there should be a logical basis to the choice of groups. In many cases, the division will be rather arbitrary and based on the preferences of the author(s). However, the following categorizations should be considered when identifying appropriate test groups within a Test Suite Structure (TSS):

- abstract architecture: A test group for each valid configuration specified. For example:
  - terminal-to-terminal direct;
  - terminal-to-terminal via a gatekeeper;
  - terminal-to-terminal via an intervening network.
- functionality: A test group for each of the major functions supported. For example:
  - basic voice call establishment;
  - basic voice call clearing;
  - supplementary service, call transfer.
- success or failure: A test group for normal behaviour and another for exceptional behaviour.

8.4.2 Define test coverage within each test group

Once a logical set of test groups has been defined, the required range of functions to be tested in each group should be specified. As an example, the coverage for a basic voice call establishment test group might include:

- successful call from User A to User B;
- successful call from User B to User A;
- unanswered call from User A to User B;
- unanswered call from User B to User A;
- call attempt from User A to a busy User B;
- call attempt from User B to a busy User A.

NOTE: In the examples above, it would be necessary to have specified the meaning of "User A" and "User B" in the context of the abstract architecture.

There should be enough information in the test coverage to ensure that tests can be specified for all of the interoperable functions of an implementation.

8.5 Write Test Purposes

Before writing the individual steps that are required to complete a test case, a full description of the objective of each test case should be specified in its Test Purpose. Without this objective, it may not be clear how the test should be defined. The following example explains the intent of the associated test case in enough detail that there should be no ambiguity for the test writer.

Test Purpose: To verify that a call can be established successfully to User B by User A and that speech communication is possible between User A and User B.

8.6 Write test cases

8.6.1 Pre-test conditions

In some instances, although not, necessarily, all, it is useful to be able to specify some pre-conditions to a test case. This often takes the form of instructions for configuring the EUT and QE to ensure that the Test Purpose is met fully. An example of a valid pre-test condition is "*Configure EUT and QE to communicate using SIP with G.711 µLaw codec*".

8.6.2 Test steps and verdicts

8.6.2.1 Test steps

Test cases describe the detailed steps that must be followed in order to achieve the stated purpose of each test. These steps should be specified in a clear and unambiguous way but without placing unreasonable restrictions on how the 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. As an example, the test step "*Pick up User A's telephone handset and dial the number of User B*" is certainly clear and unambiguous but it can only apply to a classical, physical telephone and not to a soft phone or even a mobile handset. Expressing this step as "*Initiate a new call at User A to the address of User B*" is no less clear or unambiguous but it can be applied to any type of telephone.

8.6.2.2 Verdicts

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 such as "*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. 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 EUT and QE(s) interoperate correctly, it may not be the case that a "Fail" verdict implies that they do not. The MoC plays an essential role in almost all interoperability tests but is not part of the SUT (see figure 5). A "Fail" verdict may be caused by a fault or unexpected behaviour in the MoC. Thus, each "Fail" verdict should be investigated thoroughly, possibly using monitoring equipment as shown in figure 4, to determine its root cause before either validating the verdict as a true failure (if the root cause is within the SUT) or retesting (if the root cause is determined to be outside the SUT).

8.6.2.3 Specification of test steps and verdicts

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 (e.g. "*Answer incoming call*" and "*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 (e.g. "*Cause SIP INVITE message to be sent*" or "*Was 180 Ringing received?*").

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 performed by human test drivers. This need not always be the case and in situations where automation of user functions is possible, test cases could also be written in any of the following:

- test specification languages (e.g. TTCN-3);
- programming languages (e.g. C++);
- scripting languages (e.g. PERL).

It should be noted that although test cases written only in machine-readable form offer great benefits in terms of repeatability and speed of execution, they cannot, generally, be used by human test drivers as instruction for running the tests manually. Thus, when it is not known how the tests will be performed, it is advisable to write them in a structured form of a natural language such as English.

8.6.3 Example

No assumptions should be made about the knowledge of the EUT or QE possessed by the person (or machine) carrying out the test. The sequence of actions involved in each test case should be specified in full. An example of a complete test case (including Test Purpose and pre-conditions) is shown in table 2.

Table 2: Example test case specification

Test: SS-1	**Selection Criteria:** Optional		**Selected:** Yes No
Title:	Supervised call transfer from User B to User A		
Test Purpose:	To verify that a call to User B can be transferred to User A after User B and User A have conferred together		
Pre-test conditions:	Use Test Architecture 2 Configure User A, User B and User C with Bearer Capability set to "Speech, 64 kbit/s"		

Step	**Test description**	**Verdict**	
		Pass	**Fail**
1	Initiate new call at User C to the address of User B		
2	Accept call at User B		
3	Activate the "recall" button (or equivalent) at User B's terminal		
4	*Is dial tone (or an equivalent indication) present at User B's terminal?*	Yes	No
5	Initiate a new call from User B to the address of User A		
6	*Is User A's terminal alerting (visual or audible indication)?*	Yes	No
7	Accept call at User A		
8	Apply speech at User A		
9	*Can speech from User A be heard and understood at User B?*	Yes	No
10	*Can speech from User A be heard and understood at User C?*	No	Yes
11	Apply speech at User B		
12	*Can speech from User B be heard and understood at User A?*	Yes	No
13	*Can speech from User B be heard and understood at User C?*	No	Yes
14	Clear call at User B		
15	Apply speech at User A		
16	*Can speech from User A be heard and understood at User C?*	Yes	No
17	Apply speech at User C		
18	*Can speech from User C be heard and understood at User A?*	Yes	No
19	Clear the call at User A		
20	Clear the call at User C		
Observations:			

8.6.4 Pre-amble and post-amble

In the example test case shown in table 2 it is clear that Steps 1 and 2 are essential for establishing the call and that Steps 19 and 20 are equally necessary for clearing the call but none of these steps play a significant part in the test itself as there are no verdicts associated with them. In conformance testing terminology, they can be considered to be the pre-amble (Steps 1 and 2) and the post-amble (Steps 19 and 20) and it may be useful to segregate these steps from the main testing sequence as shown by the dotted lines in table 3. Other methods of segregation (such as shading or the use of prefixes to the step numbers) are equally valid and may even be combined for greater effect.

Table 3: Test case example showing segregation of pre-amble and post-amble

Test: SS-2	**Selection Criteria:** Optional		**Selected:** Yes No
Title:	Supervised call transfer from User B to User A		
Test Purpose:	To verify that a call to User B from User C can be transferred to User A after User B and User A have conferred together		
Pre-test conditions:	Use Test Architecture 2 Configure User A, User B and User C with Bearer Capability set to "Speech, 64 kbit/s"		

Step	**Test description**	**Verdict**	
		Pass	**Fail**
P1	Initiate new call at User C to the address of User B		
P1	Accept call at User B		
3	Activate the "recall" button (or equivalent) at User B's terminal		
4	*Is dial tone (or an equivalent indication) present at User B's terminal?*	Yes	No
5	Initiate a new call from User B to the address of User A		
6	*Is User A's terminal alerting (visual or audible indication)?*	Yes	No
7	Accept call at User A		
8	Apply speech at User A		
9	*Can speech from User A be heard and understood at User B?*	Yes	No
10	*Can speech from User A be heard and understood at User C?*	No	Yes
11	Apply speech at User B		
12	*Can speech from User B be heard and understood at User A?*	Yes	No
13	*Can speech from User B be heard and understood at User C?*	No	Yes
14	Clear call at User B		
15	Apply speech at User A		
16	*Can speech from User A be heard and understood at User C?*	Yes	No
17	Apply speech at User C		
18	*Can speech from User C be heard and understood at User A?*	Yes	No
P19	Clear the call at User A		
P20	Clear the call at User C		
Observations:			

8.6.4.1 Alternative test case presentation forms

Test cases written in a structured and tabulated natural language (as in table 3) are ideal when the tests themselves are to be performed manually by human test drivers. If, however, tests are to be performed automatically using computer-based test drivers, the test cases should, perhaps, be written in an appropriate programming or scripting language. The following text shows how the example test case could be expressed in the TTCN-3 core language.

```
// Define Supervised Transfer test case
testcase SupervisedTransfer() runs on userTerminalType
{  timer ResponseTimer := 100E-3;

  // Preamble: Establish call between Users B & C
  m3s.send (CallEstablish_1);
  m2s.receive (CallEstablish_1);
  m2s.send (CallAccept_1);
  m3s.receive (CallAccept_1);

  // Register recall test
  m2s.send (Recall);
  ResponseTimer.start;
  alt
  { [] ResponseTimer.timeout
       { setverdict(fail);
```

```
     stop
   }
[] m2d.receive (DialTone)
   { setverdict(pass);
     ResponseTimer.stop

     // Hold call test
     m2s.send (CallEstablish_2);
     m1s.receive (CallEstablish_2);
     ResponseTimer.start;
     m1s.send (Alerting);
     alt
     { [] ResponseTimer.timeout
          { setverdict(fail);
            stop
          }
       [] m2s.receive (Alerting)
          { setverdict(pass);
            ResponseTimer.stop

            // Speech test 1
               m1s.send (CallAccept_2);
               m2s.receive (CallAccept_2);
               m1d.send (DTMF123456);
               ResponseTimer.start;
               alt
               { [] m3d.receive (DTMF123456)
                    { setverdict(fail);
                      stop
                    }
                 [] ResponseTimer.timeout
                    { setverdict(fail);
                      stop
                     }
                 [] m2d.receive (DTMF123456)
                    { setverdict(pass);
                      ResponseTimer.stop

                      // Speech test 2
                      m2d.send (DTMF123456);
                      ResponseTimer.start
                      alt
                      { [] m3d.receive (DTMF123456)
                           { setverdict(fail);
                             stop
                           }
                        [] ResponseTimer.timeout
                           { setverdict(fail);
                             stop
                           }
                        [] m1d.receive (DTMF123456)
                           { setverdict(pass);
                             ResponseTimer.stop

                             // Transfer test 1
                             m2s.send (CallRelease_1);
                             m1d.send (DTMF123456);
                             ResponseTimer.start;
                             alt
                             { [] ResponseTimer.timeout
                                  { setverdict(fail);
                                    stop
                                  }
                                [] m3d.receive (DTMF123456)
                                   { setverdict(pass);
                                      ResponseTimer.stop

                                      // Transfer test 2
                                      m3d.send (DTMF123456);
                                      ResponseTimer.start;
                                      alt
                                      { [] ResponseTimer.timeout
                                           { setverdict(fail);
                                             stop
                                           }
                                        [] m1d.receive (DTMF123456)
                                           { setverdict(pass);
                                             ResponseTimer.stop
```

```
                                                  // Postamble: Clear down the call
                                                  m3s.send (CallRelease_2);
                                                  m1s.send (CallRelease_2);
                                                }
   }}}}}}}}}}}}

  // The final block is the module control which initiates the
  // single defined test case.
  control
    {
          execute (SupervisedTransfer());
    }
}
```

Although the TTCN-3 core notation can be exactly and repeatedly interpreted by a suitably equipped test system, it is not so easy for a human, other than somebody skilled in the use of TTCN-3, to read and understand. If that is necessary, then the Graphical presentation Format for TTCN-3 (GFT) can be used. As an illustration, the test case defined in table 3 is shown as part of a GFT specification in figures 16 and 17.

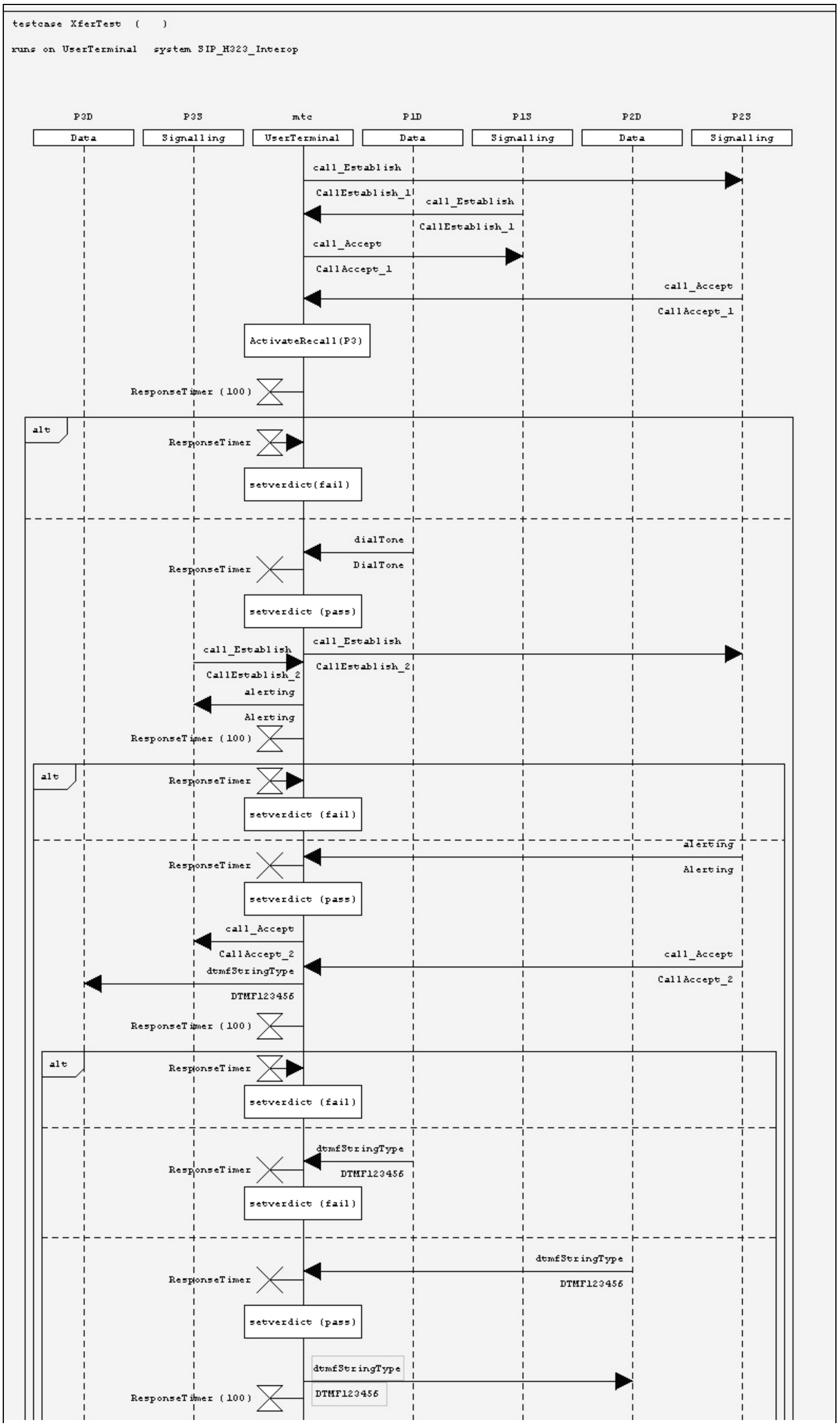


Figure 16: GFT specification of supervised transfer test case - Part 1

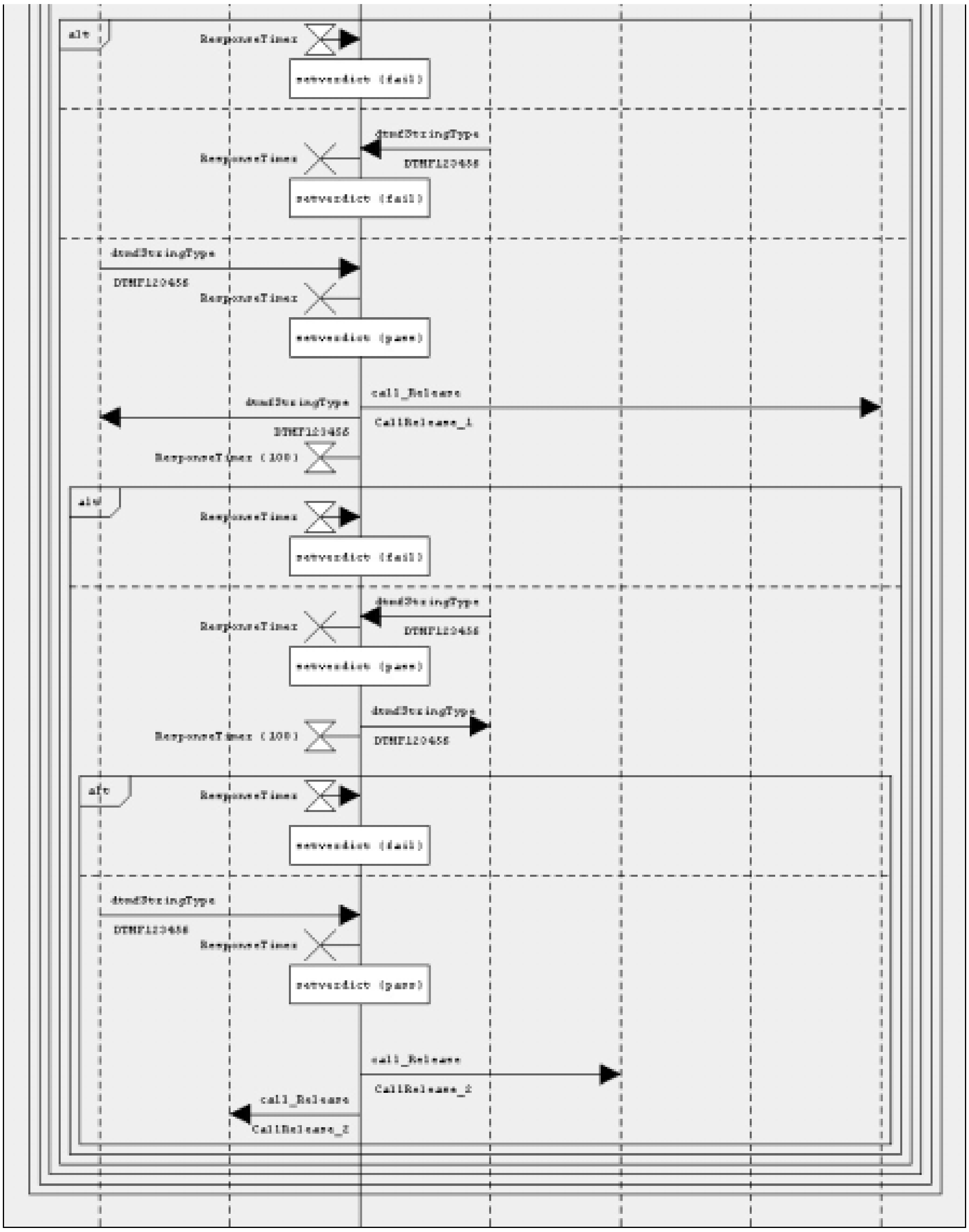


Figure 17: GFT specification of supervised transfer test case - Part 2

8.7 Validate test cases

The ideal method of validating test cases is to set up a physical test configuration and then perform each of the tests to ensure that:

- the specified pre-conditions establish the EUT and QE in the necessary configuration for the test;
- no unnecessary pre-conditions are specified;
- the individual test steps are expressed in an unambiguous way and are easy to follow;
- all necessary steps are covered from the start of the test to its completion;
- each test case fully realizes the objective of its test purpose;
- the combined intermediate and final verdicts do, in fact, lead to a true assessment of the test purpose.

In many cases, it will not be possible to validate the test cases by execution because there will not be suitable equipment available. In such situations, the simplest alternative is to carry out a structured walk-through of each test case (preferably with independent reviewers) checking every step and verdict in turn to assess the completeness and validity of the test case. Further information on walk-through and other validation methods can be found in EG 202 107 [3], "Planning for validation and testing in the standards-making process".

8.8 Finalize IFS

During the development of the Test Purposes and test cases it is possible that inconsistencies, gaps and other inaccuracies will be identified in the draft IFS. Now that the development is complete, these identified changes should be consolidated into the final IFS ready for publication.

9 Interoperability testing process

9.1 Overview

Although it is possible to automate interoperability testing, it is likely that test cases will be written in a structured natural language to be followed by human test drivers. It is, therefore, important to ensure that the defined steps and verdicts of each test case are carefully followed and recorded.

Interoperability testing involves the following three stages:

- preparing for testing;
- testing;
- writing the Test Report.

The process is expressed graphically in figure 14 using a UML activity diagram.

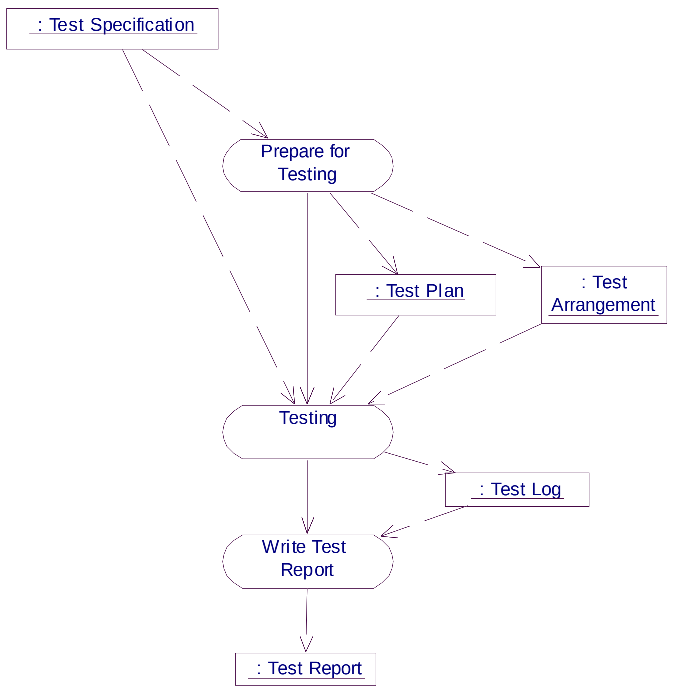


Figure 18: Interoperability testing

9.2 Prepare for testing

9.2.1 Test arrangement

Before actual testing can take place, there are a number of activities that must be completed. The first of these is to specify a test arrangement (figure 20) mapping the abstract architecture (figure 19) in the test specification to the concrete configurations that are going to be used for testing. This mapping should identify the manufacturer, product name and build status of the EUT and the QE(s). It should also specify how the various items of equipment are to be physically interconnected.

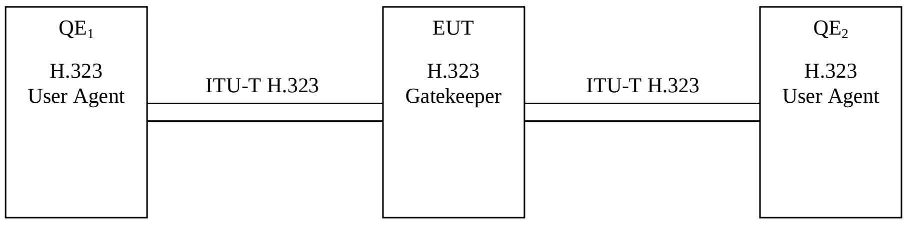


Figure 19: Example of an abstract architecture

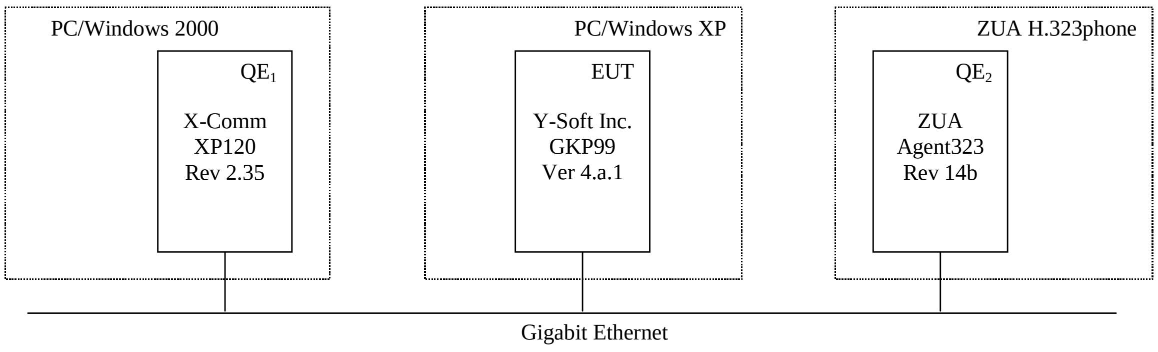


Figure 20: Test arrangement based on the example abstract architecture

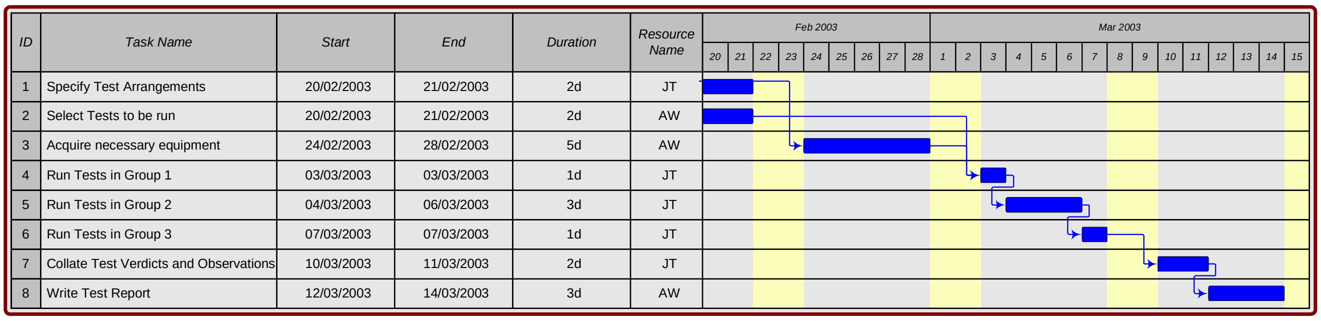
In addition to the definition of physical test arrangements, it may also be useful to specify other system configuration requirements which could include items such as the necessary numbering plan and the choice of codecs to be used in the testing.

9.2.2 Test planning

It is always advisable to take the time to prepare a plan of testing before beginning the work itself. A test plan should include:

- identification of which test cases are to be included;
- identification of which (optional) test cases are not to be included;
- indication of the order in which the tests are to be performed and the relationships between tests;
- specification of the test arrangements required for each group of tests;
- identification of equipment and facilities required to establish the necessary test configurations;
- identification of the human resources required during the testing period.

The information above should be consolidated into a formal plan against which progress can be monitored. Figure 21 shows an example test plan presented as a Gantt chart although any form of planning diagram (e.g. PERT or Timeline) could also be used.



ID	Task Name	Start	End	Duration	Resource Name
1	Specify Test Arrangements	20/02/2003	21/02/2003	2d	JT
2	Select Tests to be run	20/02/2003	21/02/2003	2d	AW
3	Acquire necessary equipment	24/02/2003	28/02/2003	5d	AW
4	Run Tests in Group 1	03/03/2003	03/03/2003	1d	JT
5	Run Tests in Group 2	04/03/2003	06/03/2003	3d	JT
6	Run Tests in Group 3	07/03/2003	07/03/2003	1d	JT
7	Collate Test Verdicts and Observations	10/03/2003	11/03/2003	2d	JT
8	Write Test Report	12/03/2003	14/03/2003	3d	AW

Tests in Group 1 use Test Configuration A
Tests in Group 2 use Test Configuration B
Tests in Group 3 use Test Configuration B

Figure 21: Example test plan

9.3 Testing

9.3.1 Manual testing

The sequence of tests specified should be grouped in a logical way that ensures efficient use of test configurations and "bottom-up" flow of tests (testing basic functionality first and then progressing to more complex functions). It is, therefore, important to carry out the tests in the sequence specified, exactly following the steps defined in each test case.

Throughout the testing process, it is essential that a record of each verdict (both intermediate and final) is kept for each test case. If the test cases are specified in a tabular, this can be used as a proforma for logging the test results. Alternatively, a simple table listing each of the test cases and their associated verdicts could be used. An example of how such a table could be constructed is shown in table 4.

Table 4: Example table summarizing test verdicts

Test case	Title	Verdict						Overall verdict	Observations
		1	2	3	4	5	6		
BS-1	Voice call establishment from User A to User B	✓	✓	✓	✓	✓	✓	Pass	
BS-2	Voice call establishment from User B to User A	✓	✓	✓	✓	✓	✓	Pass	
BS-3	Call establishment from User A to User B using en-bloc sending	×						Fail	User B's terminal failed to alert although ringing tone was heard at User A's terminal

Table 4 shows a test summary in fairly simple form. If necessary, additional information, such as a time-stamp or identification of the test driver(s), can be included if required.

9.3.2 Automated testing

If the test cases have been automated (as described in clause 8.6.2.3), the sequencing of tests and the logging of verdicts will be predetermined by the test programme. It will still be necessary to take care in establishing and modifying the test arrangements as required to ensure that the expected configurations are tested.

9.4 Write test report

A test report should summarize the testing activity and provide a clear indication of whether the tested equipment can be considered to be interoperable or not. It should include the following:

- organizational information:
  - when the testing took place;
  - where the testing took place;
  - who carried out the testing;
- equipment information:
  - test configurations used;
  - hardware and software identities for EUT and all QEs;
  - hardware and software revision states for EUT and all QEs;
  - identification of the standards (including versions) implemented in each MoC;
- testing information:
  - identification of the specific test specification upon which the testing was based;
  - identification of omitted tests (with a reason for omission if appropriate);
  - full summary of test verdicts.

Annex A (informative): Example IFS (TIPHON Profile of H.323, Release 3)

A.1 Introduction

The supplier of a protocol implementation which is claimed to conform to TS 101 883 [6] may complete the following Interoperable Features Statement (IFS) proforma if the implementation is to be submitted for interoperability testing. The IFS is a statement of which functions supported by the protocol have been implemented. The IFS can have a number of uses, including:

- as a detailed indication of the functional capabilities of the implementation;
- as a basis for initially checking the possibility of interoperating with another implementation;
- as the basis for selecting appropriate tests against which to assess the ability of the implementation to interoperate with other implementations.

A.2 Instructions for completing the IFS proforma

A.2.1 General structure of the IFS proforma

The IFS proforma is a fixed format questionnaire divided into sub-clauses each containing a group of individual items. Each item is identified by an item number, the name of the item (question to be answered), and the reference(s) to the clause(s) that specifies (specify) the item in the main body of this Standard.

The "Status" column indicates whether an item is applicable and if so whether support is mandatory or optional. The following terms are used:

M	mandatory (the function is required by TS 101 883 [6]);
O	optional (the function is not required by TS 101 883 [6], but if the function is implemented, it is required to conform to the protocol specifications);
O.<n>	optional, but support of at least one of the group of options labelled by the same numeral <n> is required;
C.<cond>	conditional requirement, depending on support for the item or items listed in condition <cond> explained below the table of appearance;
X	prohibited, the profile does not allow the support of this functionality in connection with the underlying specification, but if it is implemented it must be possible to disable (or not to enable) it;
N/A	not applicable, this feature is not contained in the profile;

References to the specification are made in the column "Reference".

Answers to the questionnaire items are to be provided either in the "Support" column, by simply marking an answer to indicate a restricted choice (Yes or No).

A.2.2 Additional information

Items of additional information allow a supplier to provide further information intended to assist the interpretation of the IFS. It is not intended or expected that a large quantity will be supplied, and a IFS can be considered complete without any such information. Examples might be an outline of the ways in which a (single) implementation can be set up to operate in a variety of environments and configurations.

References to items of additional information may be entered next to any answer in the questionnaire, and may be included in items of exception information.

A.3 IFS proforma

A.3.1 Implementation identification

Supplier	
Contact point for queries about the IFS	
Implementation name(s) and version(s) (see note)	
Other information necessary for full identification - e.g. name(s) and version(s) for machines and/or operating systems; system name(s)	
NOTE: The terms name and version should be interpreted appropriately to correspond with a suppliers terminology (e.g. type, series, model).	

A.3.2 Protocol summary

Protocol	
Protocol version	
Addenda implemented (if applicable)	
Amendments implemented	
Date of statement	

A.4 H.323 entities

Table A.1: H.323 entities

Item	H.323 entities	Reference	Support
HE1	Terminal		
HE2	Gatekeeper		
HE3	Gateway		
Comments:			

A.4.1 Roles

Table A.2: Terminal roles

Item	Role	Reference	Support
TR1	Originating terminal		
TR2	Terminating terminal		
Comments: The roles "originating" and "terminating" apply to a terminal's role regarding a call. Since a terminal is going to take each position during its usage the capabilities are not listed separately in the following clauses. If there are capabilities that apply only for one role the status field will show a "condition" that will be explained below the corresponding table.			

Table A.3: Gatekeeper roles

Item	Role	Reference	Support
GK1	Gatekeeper in serving network		
GK2	Gatekeeper in intermediate network		
GK3	Gatekeeper in home network		
Comments:			

Table A.4: Gateway roles

Item	Role	Reference	Support
GW1	Originating Gateway		
GW2	Terminating Gateway		
Comments: The roles "originating" and "terminating" apply to a gateway's role regarding a call. Since a gateway is going to take each position during its usage the capabilities are not listed separately in the corresponding clauses. If there are capabilities that apply only for one role the status field will show a "condition" that will be explained below the corresponding table.			

A.4.2 Terminal capabilities

A.4.2.1 Gatekeeper discovery

Table A.5: Terminal gatekeeper discovery capabilities

Item	Capabilities	Reference	Status	Support
T_GKD1	Automatic multicast	[6] 5.1	M	
T_GKD2	Unicast	[6] 5.1	O	
T_GKD3	Authentication method negotiation	[6] 5.1.1.1	O	
Comments:				

A.4.2.2 Registration

Table A.6: Terminal registration capabilities

Item	Function	Reference	Status	Support
T_REG1	Registration without authentication	[6] 5.2	M	
T_REG2	Registration with authentication	[6] 5.2.1.1	C.1	
T_REG3	Cancelling a registration	[6] 5.3.1	M	
T_REG4	Registration update/Lightweight RRQ	[6] 5.4.1	M	
T_REG5	Additive registration	[6] 5.2.1	X	
Comments:				
C.1: *If* T_GKD3 *then* M *else* N/A.				

A.4.2.3 Basic call

Table A.7: Terminal basic call capabilities

Item	Function	Reference	Status	Support
T_BC1	PregrantedARQ	[6] 5.2.2.1.1	O.1	
T_BC2	Call admission (ARQ)	[6] B.1.2.5	O.1	
T_BC3	Fast connect procedure	[6] 6.2.1	M	
T_BC4	H.245 Tunnelling	[6] 6.2.1	M	
T_BC5	Overlap sending	[6] 6.2.1.1	M	
T_BC6	En bloc procedure	[6] 6.2.1.2	M	
T_BC7	Call release	[5] 8.5	M	
Comments: T_BC5 is only relevant for the functional entity of an originating terminal.				

A.4.2.4 Supplementary services

Table A.8: Terminal supplementary services capabilities

Item	Function	Reference	Status	Support
T_SS1	CLIP	[6] 8.1	O	
Comments:				

A.4.3 Gatekeeper capabilities

A.4.3.1 Gatekeeper in the serving network

A.4.3.1.1 Overall capabilities

Table A.9: Serving gatekeeper overall capabilities

Item	Capabilities	Reference	Status	Support
S_OC1	Gatekeeper routed model		M	
Comments:				

A.4.3.1.2 Gatekeeper discovery

Table A.10: Serving gatekeeper discovery capabilities

Item	Capabilities	Reference	Status	Support
S_GKD1	Automatic multicast	[6] 5.1	M	
S_GKD2	Unicast	[6] 5.1	M	
Comments:				

A.4.3.1.3 Registration

Table A.11: Serving gatekeeper registration capabilities

Item	Function	Reference	Status	Support
S_REG1	Registration	[6] 5.2.2.2	M	
S_REG3	Cancelling a registration	[6] 5.3.2.2	M	
S_REG4	Registration update/Lightweight RRQ	[6] 5.4.3	M	
Comments:				

A.4.3.1.4 Basic call

Table A.12: Serving gatekeeper basic call capabilities

Item	Function	Reference	Status	Support
S_BC1	PregrantedARQ	[6] 5.2.2.1.1	O.1	
S_BC2	Call admission (ARQ)	[6] B.1.2.5	O.1	
S_BC3	Fast connect procedure	[6] 6.3.1	M	
S_BC4	H.245 Tunnelling	[6] 6.3.1	M	
S_BC5	Overlap sending	[6] 6.3.1	M	
S_BC6	En bloc procedure	[6] 6.3.1	M	
S_BC7	Call release	[5] 8.5	M	
Comments:				

A.4.3.1.5 Supplementary services

Table A.13: Serving gatekeeper supplementary services capabilities

Item	Function	Reference	Status	Support
S_SS1	CLIP	[6] 8.2	O	
Comments:				

A.4.3.2 Gatekeeper in the intermediate network

A.4.3.2.1 Overall capabilities

Table A.14: Intermediate gatekeeper overall capabilities

Item	Capabilities	Reference	Status	Support
I_OC1	Gatekeeper routed model		M	
Comments:				

A.4.3.2.2 Gatekeeper discovery

Table A.15: Intermediate gatekeeper discovery capabilities

Item	Capabilities	Reference	Status	Support
I_GKD1	Unicast	[6] 5.1.5.1	M	
Comments:				

A.4.3.2.3 Registration

Table A.16: Intermediate gatekeeper registration capabilities

Item	Function	Reference	Status	Support
I_REG1	Registration	[6] 5.2.2	M	
I_REG2	Cancelling a registration	[6] 5.3.2.2	M	
I_REG3	Registration update/Lightweight RRQ	[6] 5.4.2	M	
Comments:				

A.4.3.2.4 Basic call

Table A.17: Intermediate gatekeeper basic call capabilities

Item	Function	Reference	Status	Support
I_BC1	Pregranted ARQ	[6] B.1.2.5	M	
I_BC2	Fast connect procedure	[6] 6.3.1	M	
I_BC3	H.245 Tunnelling	[6] 6.3.1	M	
I_BC4	En bloc procedure	[6] 6.3.1	M	
I_BC5	Overlap sending	[6] 6.3.1	M	
I_BC6	Call release	[5] 8.5	M	
Comments:				

A.4.3.2.5 Supplementary services

Table A.18: Intermediate gatekeeper supplementary services capabilities

Item	Function	Reference	Status	Support
I_SS1	CLIP	[6] 8.2	O	
Comments:				

A.4.3.3 Gatekeeper in the home network

A.4.3.3.1 Overall capabilities

Table A.19: Home gatekeeper overall capabilities

Item	Capabilities	Reference	Status	Support
H_OC1	Gatekeeper routed model	[6] 6.4.1	M	
Comments:				

A.4.3.3.2 Gatekeeper discovery

Table A.20: Home gatekeeper discovery capabilities

Item	Capabilities	Reference	Status	Support
H_GKD1	Unicast	[6] 5.1	M	
H_GKD2	Authentication method negotiation	[6] 5.1.3	O	
Comments:				

A.4.3.3.3 Registration

Table A.21: Home gatekeeper registration capabilities

Item	Function	Reference	Status	Support
H_REG1	Registration without authentication	[6] 5.2.2	M	
H_REG2	Registration with authentication	[6] 5.2.2.1.1	M	
H_REG3	Cancelling a registration	[6] 5.3.2.2	M	
H_REG4	Registration update/Lightweight RRQ	[6] 5.4.2	M	
H_REG5	Additive registration	[6] 5.2.2.1.2	X	
Comments:				

A.4.3.3.4 Basic call

Table A.22: Home gatekeeper basic call capabilities

Item	Function	Reference	Status	Support
H_BC1	PregrantedARQ	[6] 5.2.2.1.1	M	
H_BC2	Fast connect procedure	[6] 6.4.1	M	
H_BC3	H.245 Tunnelling	[6] 6.4.1	M	
H_BC4	En bloc procedure	[6] 6.4.1.2	M	
H_BC5	Overlap sending	[6] 6.4.1.1	M	
H_BC6	Call release	[5] 8.5	M	
Comments:				

A.4.3.3.5 Supplementary services

Table A.23: Home gatekeeper supplementary services capabilities

Item	Function	Reference	Status	Support
H_SS1	CLIP	[6] 8.2	O	
H_SS2	CLIR	[6] 8.2	O	
Comments:				

A.4.4 Gateway capabilities

A.4.4.1 Overall capabilities

Table A.24: Gateway overall capabilities

Item	Capabilities	Reference	Status	Support
W_OC1	Gatekeeper routed model	[6] 6.2.1	M	
Comments:				

A.4.4.2 Basic call

Table A.25: Gateway basic call capabilities

Item	Function	Reference	Status	Support
W_BC1	PregrantedARQ	[6] 5.2.2.1.1	O.1	
W_BC2	Call admission (ARQ)	[6] B.1.2.5	O.1	
W_BC3	Fast connect procedure	[6] 6.5.1	M	
W_BC4	H.245 Tunnelling	[6] 6.5.1	M	
W_BC5	Overlap sending	[6] 6.5.1.1	C.1	
W_BC6	En bloc procedure	[6] 6.5.1.2	M	
W_BC7	Call release	[5] 8.5	M	
Comments:				
C.1: *if* GW1 *then* M *else* N/A.				

A.4.4.3 Supplementary services

Table A.26: Gateway supplementary services capabilities

Item	Function	Reference	Status	Support
W_SS1	CLIP	[6] 8.3	O	
W_SS2	CLIR	[6] 8.3	O	
Comments: W_SS1 and W_SS2 refer to correct forwarding of the respective parameters.				

Annex B (informative): Example IFS (TIPHON Profile of SIP, Release 3)

B.1 Introduction

The supplier of a protocol implementation which is claimed to conform to TS 101 884 [7] may complete the following Interoperable Features Statement (IFS) proforma if the implementation is to be submitted for interoperability testing. The IFS is a statement of which functions supported by the protocol have been implemented. The IFS can have a number of uses, including:

- as a detailed indication of the functional capabilities of the implementation;
- as a basis for initially checking the possibility of interoperating with another implementation
- as the basis for selecting appropriate tests against which to assess the ability of the implementation to interoperate with other implementations.

B.2 Instructions for completing the IFS proforma

B.2.1 General structure of the IFS proforma

The IFS proforma is a fixed format questionnaire divided into sub-clauses each containing a group of individual items. Each item is identified by an item number, the name of the item (question to be answered), and the reference(s) to the clause(s) that specifies (specify) the item in the main body of this Standard.

The "Status" column indicates whether an item is applicable and if so whether support is mandatory or optional. The following terms are used:

M	mandatory (the function is required by TS 101 884 [7]);
O	optional (the function is not required by TS 101 884 [7], but if the function is implemented, it is required to conform to the protocol specifications);
O.<n>	optional, but support of at least one of the group of options labelled by the same numeral <n> is required;
C.<cond>	conditional requirement, depending on support for the item or items listed in condition <cond> explained below the table of appearance;
N/A	not applicable, this feature is not contained in the profile;

References to the specification are made in the column "Reference".

Answers to the questionnaire items are to be provided either in the "Support" column, by simply marking an answer to indicate a restricted choice (Yes or No), or in the "Not Applicable" column (N/A).

B.2.2 Additional information

Items of additional information allow a supplier to provide further information intended to assist the interpretation of the IFS. It is not intended or expected that a large quantity will be supplied, and a IFS can be considered complete without any such information. Examples might be an outline of the ways in which a (single) implementation can be set up to operate in a variety of environments and configurations.

References to items of additional information may be entered next to any answer in the questionnaire, and may be included in items of exception information.

B.3 IFS proforma

B.3.1 Implementation identification

Supplier	
Contact point for queries about the IFS	
Implementation name(s) and version(s) (see note)	
Other information necessary for full identification - e.g. name(s) and version(s) for machines and/or operating systems; system name(s)	
NOTE: The terms name and version should be interpreted appropriately to correspond with a suppliers terminology (e.g. type, series, model).	

B.3.2 Protocol Summary, EN 301 xxx

Protocol version	
Addenda implemented (if applicable)	
Amendments implemented	
Date of statement	

B.4 SIP entities

Table B.1: SIP entities

Item	SIP entities	Reference	Support
SE1	User agent		
SE2	Registrar		
SE3	Proxy		
SE4	Gateway		
Comments:			

B.4.1 Roles

Table B.2: User agent roles

Item	Role	Reference	Support
UA1	Originating user agent		
UA2	Terminating user agent		
Comments: The roles "originating" and "terminating" apply to a User Agent's role regarding a call. Since a user agent is going to take each position during its usage the capabilities are not listed separately in the following clauses. If there are capabilities that apply only for one role the status field will show a "condition" that will be explained below the corresponding table.			

Table B.3: Registrar roles

Item	Role	Reference	Support
RE1	Registrar in the home network		
Comments:			

Table B.4: Proxy roles

Item	Role	Reference	Support
PR1	Proxy in serving network		
PR2	Proxy in intermediate network		
PR3	Proxy in home network		
Comments:			

Table B.5: Gateway roles

Item	Role	Reference	Support
GW1	Originating gateway		
GW2	Terminating gateway		
Comments: The roles "originating" and "terminating" apply to a gateway's role regarding a call. Since a gateway is going to take each position during its usage the capabilities are not listed separately in the corresponding clauses. If there are capabilities that apply only for one role the status field will show a "condition" that will be explained below the corresponding table.			

B.4.2 User Agent capabilities

B.4.2.1 Registration

Table B.6: User Agent registration capabilities

Item	Function	Reference	Status	Support
U_REG1	Unicast registration	[7] 5.1.1	M	
U_REG2	Multicast registration	[2]10.2.6	O	
U_REG3	Authenticated registration	[7] 5.1.1.1	M	
U_REG3	Additive registration	[7] 5.1.1.1	M	
U_REG4	Refreshing contact addresses	[7] 5.2.1	M	
U_REG5	Removing contact addresses/Deregistration	[7] 5.3.1	M	
Comments:				

B.4.2.2 Basic call

Table B.7: User agent basic call capabilities

Item	Function	Reference	Status	Support
U_BC1	Call establishment without authentication	[7] 5.2.2.1.1	M	
U_BC2	Call establishment with authentication	[7] 6.2.1	O	
U_BC3	Call clearing of an active call	[7] 6.2.1	M	
U_BC4	Call clearing before destination answers	[7] 6.2.1	M	
U_BC5	Rejection of incoming call	[7] 6.2.1.1	M	
U_BC6	Call clearing authenticated	[7] 6.2.1.2	M	
Comments:				

B.4.3 Registrar capabilities

B.4.3.1 Registration

Table B.8: Registrar capabilities

Item	Function	Reference	Status	Support
U_REG1	Unicast registration	[7] 5.1.1	M	
U_REG2	Multicast registration	[2] 10.2.6	O	
U_REG3	Authenticated registration	[7] 5.1.2.1.1	M	
U_REG4	Additive registration	[7] 5.1.1.1	M	
U_REG5	Refreshing contact addresses	[7] 5.2.2	M	
U_REG6	Removing contact addresses/Deregistration	[7] 5.3.2	M	
Comments:				

B.4.4 Proxy capabilities

B.4.4.1 Proxy in the serving and intermediate network

B.4.4.1.1 Registration

Table B.9: Serving/Intermediate proxy registration capabilities

Item	Function	Reference	Status	Support
S_REG1	Unicast registration	[7] 5.1.2	M	
S_REG2	Multicast registration	[2] 10.2.6	C.1	
S_REG3	Additive registration	[7] 5.1.1.1	M	
S_REG4	Refreshing contact addresses	[7] 5.2.3	M	
S_REG5	Removing contact addresses/Deregistration	[7] 5.3.3	M	
Comments:				
C.1: *if* PR1 *then* M *else* N/A.				

B.4.4.1.2 Basic call

Table B.10: Serving/Intermediate proxy basic call capabilities

Item	Function	Reference	Status	Support
S_BC1	Call establishment without authentication	[7] 6.3.1	M	
S_BC2	Call clearing of an active call	[7] 6.3.1	M	
S_BC3	Call clearing before destination answers	[7] 6.3.1	M	
Comments:				

B.4.4.2 Proxy in the home network

B.4.4.2.1 Registration

Table B.11: Home proxy registration capabilities

Item	Function	Reference	Status	Support
H_REG1	Unicast registration	[7] 5.1.2	M	
H_REG3	Additive registration	[7] 5.1.1.1	M	
H_REG4	Refreshing contact addresses	[7] 5.2.2	M	
H_REG5	Removing contact addresses/Deregistration	[7] 5.3.3	M	
Comments:				

B.4.4.2.2 Basic call

Table B.12: Home proxy basic call capabilities

Item	Function	Reference	Status	Support
H_BC1	Call establishment with authentication	[7] 6.4.1	M	
H_BC2	Call clearing of an active call	[7] 6.4.2	M	
H_BC3	Call clearing before destination answers	[7] 6.4.2	M	
H_BC4	Call clearing authenticated	[7] 6.4.2	M	
Comments:				

B.4.5 Gateway capabilities

B.4.5.1 Basic call

Table B.13: User agent basic call capabilities

Item	Function	Reference	Status	Support
G_BC1	Call establishment without authentication	[7] 5.2.2.1.1	M	
G_BC2	Call establishment with authentication	[7] 6.10.1	C.1	
G_BC3	Call clearing of an active call	[7] 6.9/6.10.2	M	
G_BC4	Call clearing before destination answers	[7] 6.10.2	C.2	
Comments:				
C.1: *if* GW1 *then* O *else* N/A. C.2: *if* GW1 *then* M *else* N/A.				

Annex C (informative): Experimental testbed

C.1 Description of an experimental TIPHON H.323 - TIPHON SIP interoperability testbed

As part of the validation of the guidelines specified in the present document, an experimental testing facility ("testbed") was established and a range of interoperability tests were performed. The testbed configuration linked together H.323 and SIP equipment and used the TIPHON H.323 – SIP interoperability tests specified in TS 102 237-2 [4].

NOTE: At the time of the writing TS 102 237-2 [4], the profiles for TIPHON Release 4 were not available. Hence, this interoperability test specification is based on Release 3.

The requirements specified for the testbed were as follows:

- the products to be tested may either be software (SW) applications or stand-alone hardware (HW) devices;
- the interoperability testing should be conducted according to the methodology described in the present document. This includes the creation of the interoperability testing documentation, the test planning as well as the test execution and the test reporting. If available, pro-forma interoperability testing documentation should be used;
- future versions may extend the interoperability testbed definition to include 3GPP SIP.

Additional, more detailed requirements were specified with respect to the following:

- interoperability testing documentation;
- IP infrastructure;
- VoIP applications;
- monitoring facilities.

C.1.1 TIPHON H.323 - TIPHON SIP interoperability testing documentation

As clause 9 of the present document outlines, the three stages of the interoperability testing process are:

- preparing for testing;
- testing;
- test reporting.

The testing documentation may be similarly categorized and figure 18 shows the interrelation between the testing documentation and the stages of the testing process.

Documentation related to the preparation phase is:

- Test specification

  contains the IFS(s), the abstract architecture, TSS, TP and the test cases. This documentation is independent from the testbed. It is derived from the base standard(s) and can be found in:

  - TIPHON H.323 IFS, Release 3, (annex A);
  - TIPHON SIP IFS, Release 3, (annex B);
  - TIPHON H.323 – TIPHON SIP interoperability test scenarios, TS 102 237-2 [4] which includes the abstract architecture, TSS & TP and the test cases;

- Test plan

  contains the architecture of the testbed as well as organizational issues such as the selection of the test cases, the required infrastructure (here, IP infrastructure), time schedule and human resources.

During the testing phase, the primary documentation produced is logging information related to the individual tests themselves.

The final test report contains the results of the testing and, for this purpose, pro-forma reports may also be developed. Clause 9.4 describes in more detail what information should be in a test report.

C.1.2 IP infrastructure

The required IP network was determined by mapping the abstract architecture to the physical architecture considering the needs of the individual components. As a single IP infrastructure was unlikely to meet all possible scenarios, it was necessary to consider the following aspects when defining the IP infrastructure of the testbed:

- division of the network into IP sub-networks;
- IP address ranges;
- amount of public IP addresses;
- required IP services, such as:
  - DHCP;
  - DNS;
  - NAT
- security services such as firewalls.

The next step was to set up the infrastructure and implement the IP services defined in the previous analysis. The following list identifies hardware components that are likely to appear in almost any testbed:

- routers;
- switches;
- hubs;
- connectors, such as:
  - RJ-45 connectors;
  - RJ-11 connectors e.g. for telephones and terminal adaptors (TAs);
- Hosts for running the IP services DHCP, DNS, NAT, firewall.

It is worth noting that when setting up an interoperability testbed, the actual devices used may not be determined solely by the requirements of the testbed but also by the available hardware. For example, if a router including NAT and a firewall is not available, an application gateway could be used instead. Figure C.1 illustrates a possible set up for a testbed including all HW components. It depicts the physical architecture for an Administrative Domain (AD) that also allows calls entering from the public Internet. The HW-phone and an application in the Laptop are the terminals and a desktop PC application acts as a server for DNS, DHCP and a gatekeeper.

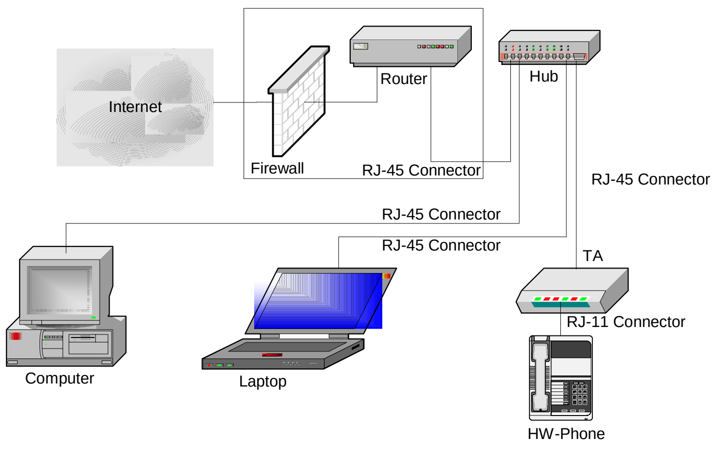


Figure C.1: An example testbed set up

The power supply is a further important consideration in testbeds that are reasonably extensive. This is especially true for large interoperability events. Thus, the power consumption of each of the interconnected devices may be important.

C.1.3 Requirements for VoIP applications

Both the IP services and the applications themselves may require additional devices and applications. The TIPHON H.323 – TIPHON SIP interoperability testbed includes VoIP applications that are illustrated in the following two lists.

Applications defined in the TIPHON H.323 profile:

- Terminal;
- Gatekeeper;
- Gateway.

Applications defined in the TIPHON SIP profile:

- User Agent;
- Proxy;
- Registrar;
- Gateway.

Generally, the requirements of these VoIP applications can be divided into two categories:

- requirements that have implications for the infrastructure, for example:
  - HW for VoIP applications to run;
  - standard telephone sets to be plugged into any TAs;

- general requirements relating to the framework for the testing itself, for example:
  - a dialling plan;
  - a definition of the codecs to be used.

C.1.4 Monitoring facilities

Monitoring facilities are neither a matter of standardization nor a direct part of the methodology. However it is sensible to use such facilities since they ease error tracking. In interoperability testing where many applications often interact with each other, error tracking may not be intuitive and the use of monitoring equipment may be particularly useful. The extent to which monitoring equipment is used and the types of monitors themselves will depend on the architecture of the testbed and the individual devices. Many applications have logging features already implemented.

Figure C.2 shows another example testbed including monitoring equipment. Note, that the logging facilities do not have to be set up on separate machines. They may be collocated with other services on a PC.

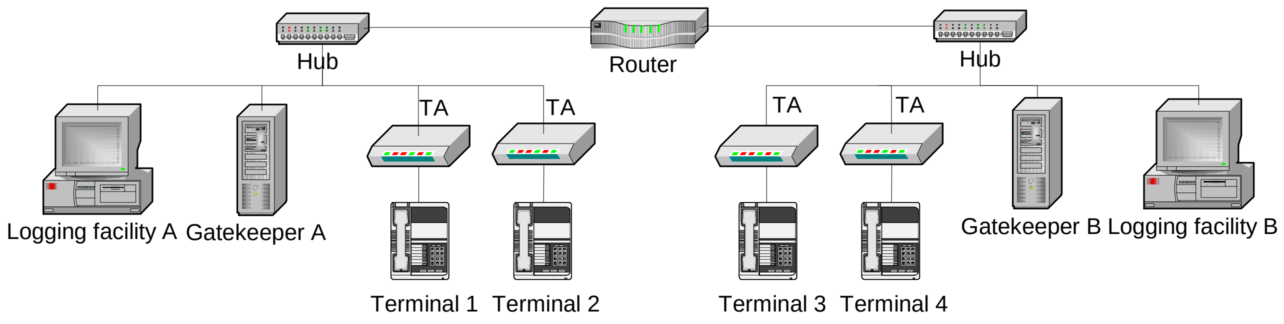


Figure C.2: Example testbed including monitoring facilities

C.2 The experimental H.323-SIP testbed

During the development of the methodology an experimental testbed was set up. The purpose of this testbed was to gather practical experience with interoperability testing and to feedback experiences into the standardization work.

C.2.1 Components of the interoperability testbed

The individual VoIP applications which were the essential components of the testbed were either supplied, free of charge, by product vendors or were obtained as open source products.

C.2.1.1 Supporting companies

The support of the industry was crucial for the success of the testbed and the following four companies generously provided commercial VoIP SW and HW for the duration of the experiment:

- L&T Infotech;
- I3Micro;
- SIPComm;
- BearingPoint INFONOVA.

C.2.1.2 Open source software

The use of Open Source software leads to a diversification of test scenarios and to test results with enhanced significance for both the validation of the H.323-SIP interoperability test suite and feedback to the participating companies.

The following open source products were used within the testbed experiment:

- Netmeeting, H.323 terminal element;
- OpenPhone, H.323 terminal element;
- OpenGK, H.323 gatekeeper;
- Windows Messenger 4.6, SIP terminal element;
- Ubiquity, SIP terminal element;
- SIPTREX, SIP terminal element;
- SIPTREX, SIP Proxy.

C.2.2 Interoperability testing activities

In order to minimize the amount of resource spent on testing, a "bottom-up" approach was taken with the following activities carried out in the order specified here:

1) Basic Interconnection Testing (BIT);
2) Interoperability testing for SIP and H.323 devices separately;
3) H.323-SIP (interworking) interoperability testing.

Taking this approach meant that known reasons for unexpected behaviour in a former testing session helped to locate deviations from expected behaviour in a subsequent one.

Although the purpose of BIT is to test direct calls between end points and the registration of end points at gatekeepers and servers, it had the side benefit that it provided a good process of familiarization with each of the products installed in the testbed.

H.323 interoperability testing and SIP interoperability testing was carried out mainly to provide feedback to the supporting vendors on the behaviour of their products.

The H.323-SIP interoperability testing was conducted according to developed methods based on the generic methodology. It could have been run without previously executing the individual H.323 and SIP interoperability tests. However, prior knowledge of the behaviour of each of the installed devices helped to locate the cause of test case failures and thus reduced the amount of resource spent. Although recommended by the methodology in the present document, it was not possible to carry out conformance testing on any of the products as there is no appropriate test equipment available for this purpose.

C.2.3 Interoperability test specification documentation

Clause 8 identifies the following documentation as essential parts on an interoperability test specification:

- Interoperable Features Statement (IFS);
- Test Suite Structure (TSS);
- Test Purposes (TP);
- Test Suite.

TS 102 237-2 [4] incorporates TSS, TPs and the Test Suite and was used as the basis for testing. H.323 and SIP IFS documents were not available and were developed as part of the interoperability testing experiment. They can be found in annexes A and B respectively.

C.2.4 Abstract architecture

Figure C.3 illustrates the abstract architecture for H.323-SIP interoperability testing. It shows the IWF as well as the elements using H.323 or SIP as MoC.

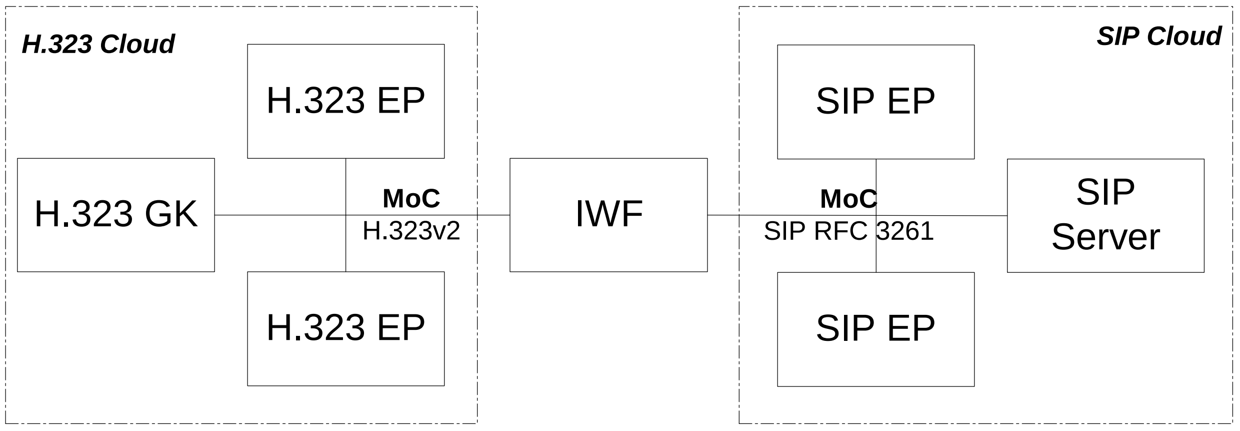


Figure C.3: Abstract architecture of the H.323-SIP interoperability testbed

C.2.5 Testing arrangements

C.2.5.1 BIT

The three test arrangement for BIT are shown in figure C.4 (end-to-end communication), figure C.5 (H.323 registration) and figure C.6 (SIP registration). In each arrangement, the QE is an EP and the EUT is an EP, a H.323 gatekeeper or a SIP server.

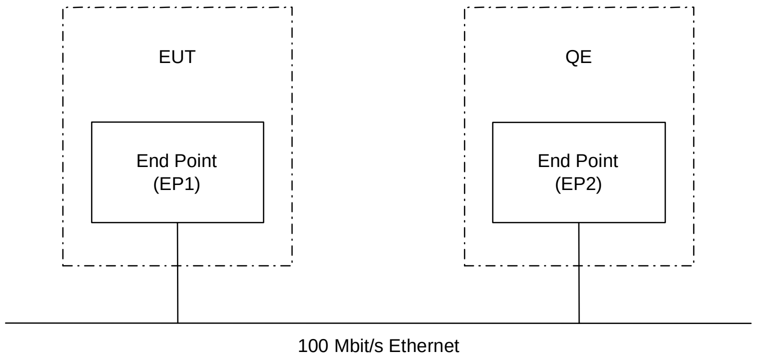


Figure C.4: Basic interconnection test - end-to-end communication

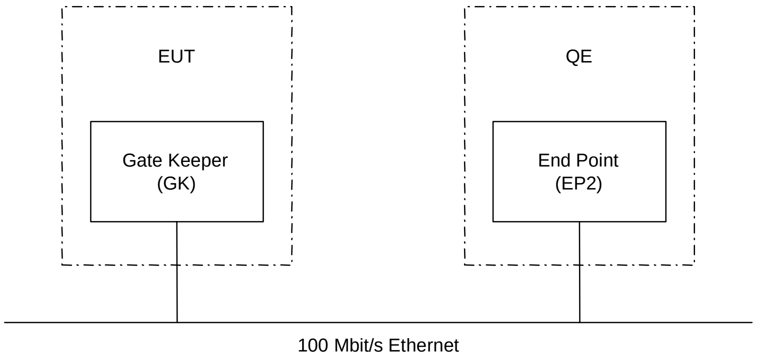


Figure C.5: Basic interconnection test - H.323 registration

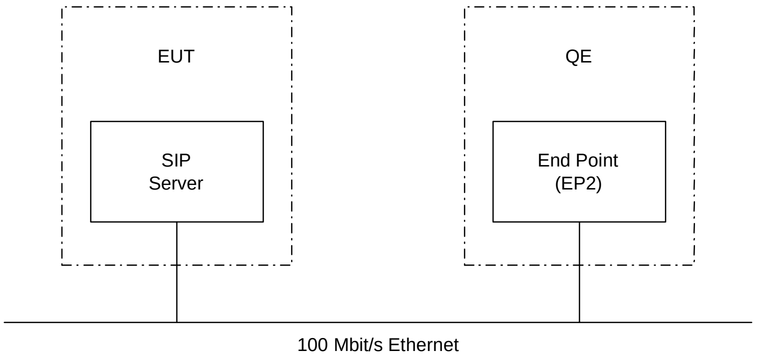


Figure C.6: Basic interconnection test - SIP registration

C.2.5.2 H.323 interoperability testing and SIP interoperability testing

The test arrangements shown in figures C.7, C.8, C.9 and C.10 were used for the testing of EPs, H.323 Gatekeepers and SIP Servers for simple interoperability without interworking.

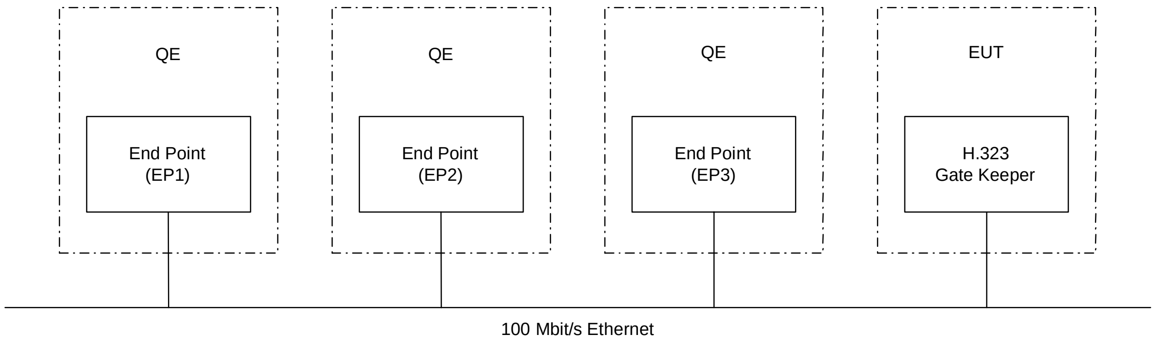


Figure C.7: H.323 interoperability testing without interworking (Gatekeeper EUT)

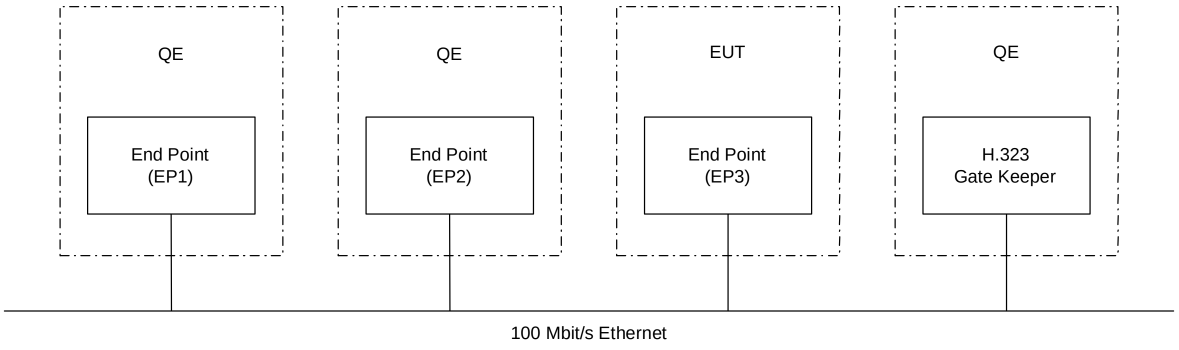


Figure C.8: H.323 interoperability testing without interworking (End point EUT)

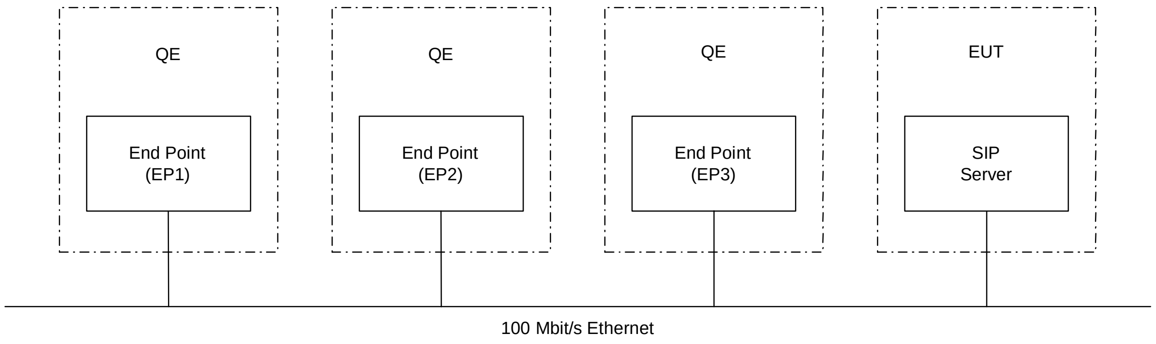


Figure C.9: SIP interoperability testing without interworking (Server EUT)

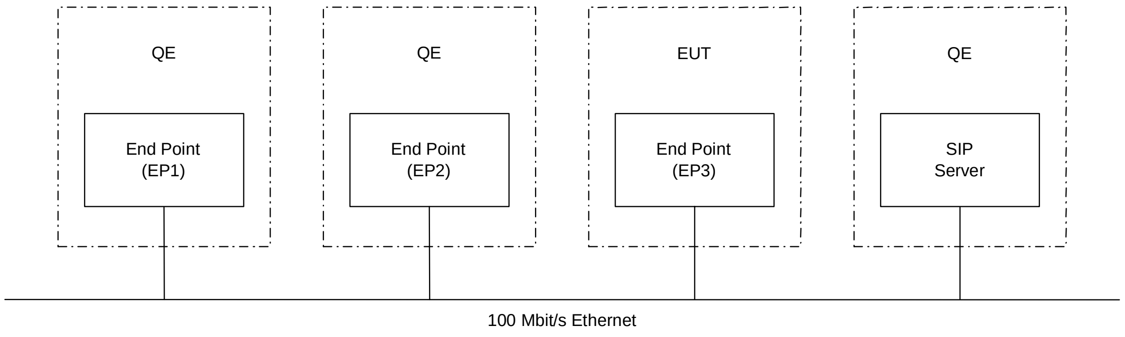


Figure C.10: SIP interoperability testing without interworking (End point EUT)

C.2.5.3 H.323-SIP interoperability testing

The test arrangement used for the H.323-SIP interworking testing through an interworking function is shown in figure C.11

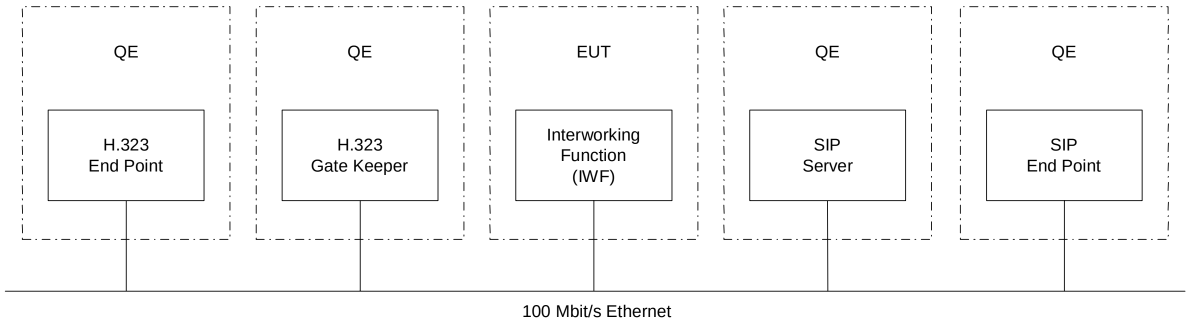


Figure C.11: H.323-SIP interoperability testing

C.2.6 Logging

For the purpose of logging, the freeware software tool, "Ethereal", including its H.323 plug-in was used. Further logging information was obtained through product-internal logging facilities.

C.2.7 Example result matrices

Table C.1 shows an example result matrix for the registration of terminals. The composition of the matrix shows that the results of all terminals trying the same functionality at each gatekeeper. The advantage of this structure is that it simplifies the comparison of results between all available devices. The disadvantage is that this kind of illustration works only if the number of test cases is small. Even with this relatively simple test configuration it is not possible to enter very detailed comments.

Table C.1: Example result matrix for EP registration

			Reference	**Gatekeeper**			**Comments**
				GK1	**GK2**	**GK3**	
Terminals	**EP1**	Manual GK discovery and registration	A.4.2.1.1	PASS	PASS	PASS	
		Automatic GK discovery and registration	A.4.2.1.2	FAIL	FAIL	FAIL	EP feature not available
		Registration update	A.4.5.1.2	PASS	PASS	PASS	
		Deregistration	A.4.2.3.1	PASS	PASS	PASS	
	EP2	Manual GK discovery and registration	A.4.2.1.1	PASS	PASS	PASS	
		Automatic GK discovery and registration	A.4.2.1.2	PASS	PASS	PASS	
		Registration update	A.4.5.1.2	PASS	PASS	PASS	
		Deregistration	A.4.2.3.1	PASS	PASS	PASS	
	EP3	Manual GK discovery and registration	A.4.2.1.1	PASS	PASS	PASS	
		Automatic GK discovery and registration	A.4.2.1.2	FAIL	FAIL	FAIL	EP feature not available
		Registration update	A.4.5.1.2	PASS	PASS	PASS	
		Deregistration	A.4.2.3.1	PASS	PASS	PASS	
	EP4	Manual GK discovery and registration	A.4.2.1.1	PASS	PASS	PASS	
		Automatic GK discovery and registration	A.4.2.1.2	PASS	PASS	PASS	
		Registration update	A.4.5.1.2	PASS	PASS	PASS	
		Deregistration	A.4.2.3.1	PASS	PASS	PASS	
	EP5	Manual GK discovery and registration	A.4.2.2.1	PASS	PASS	PASS	
		Automatic GK discovery and registration	A.4.2.2.2	FAIL	FAIL	FAIL	EP feature not available
		Registration update	A.4.5.1.2	PASS	PASS	PASS	
		Deregistration	A.4.2.3.1	FAIL	FAIL	FAIL	EP feature not available

A further example of a result matrix is illustrated in table C.2. This matrix is the outcome of one testing session. It lists the test cases with all their results.

Table C.2: Example result matrix for H.323-SIP interoperability testing

H.323-SIP interoperability testing - Test Result Matrix			
EUT:	**H.323 Gatekeeper B**		
QE:	**H.323 Endpoint A H.323-SIP IWF A SIP Proxy B SIP Endpoint A**		
Test case reference	**Test case number**	**Overall Verdict**	**Comment**
8.2.1.3	RIS_03	PASS	
8.2.1.4	RIS_04	PASS	
8.2.3.2	DR_02	PASS	
8.3.1.1	CEE_01	PASS	
8.3.1.2	CEE_02	PASS	
8.3.3.1	CC_01	PASS	
8.3.3.2	CC_02	PASS	
8.3.3.3	CC_03	PASS	
8.3.3.4	CC_04	PASS	
8.3.3.5	CC_05	PASS	
8.3.3.6	CC_06	PASS	
8.3.4.1	CPB_01	PASS	
8.3.4.2	CPB_02	PASS	
8.3.5.1	CPNA_01	FAIL	No alerting timeout implemented
8.3.5.2	CPNA_02	FAIL	No alerting timeout implemented

C.2.8 Reporting

Reports have been produced for all different testing activities except BIT as this phase of testing was only used to establish and prove the general operation of the equipment.

Since the test reports contain sensitive technical information about the products involved in the experiment and ETSI is bound by NDAs to protect the interests of the vendors, these reports have not been included in the present document.

There is no intention to make the test reports public. They have been used to improve the quality of the test suites [4] as well as to validate the generic methodology. They have also been sent to each of the supporting vendors as feedback for their products in return to their support.

Annex D (informative): Bibliography

Sawai, K.; Konno, T.; Hatafuku, M.; Gotoh, K.; Suzuki, S.; Kazama, K.: "Interconnectability testing method-AICTS", Networks, 1995. Theme: Electrotechnology 2000: Communications and Networks. [in conjunction with the] International Conference on Information Engineering., Proceedings of IEEE Singapore International Conference on , 1995 Page(s): 289 - 293

Sungwon Kang: "Relating interoperability testing with conformance testing",Global Telecommunications Conference, 1998. GLOBECOM 1998. The Bridge to Global Integration. IEEE, Volume: 6, 1998 Page(s): 3768 -3773 vol.6

Griffeth, N.; Hao, R.; Lee, D.; Sinha, R.K.: "Interoperability testing of VoIP systems", Global Telecommunications Conference, 2000. GLOBECOM '00. IEEE, Volume: 3, 2000 Page(s): 1565 -1570 vol.3

"The Authoritative Dictionary of IEEE Standards Terms", Seventh Edition, IEEE 100.

History

Document history		
V4.1.1	December 2003	Publication