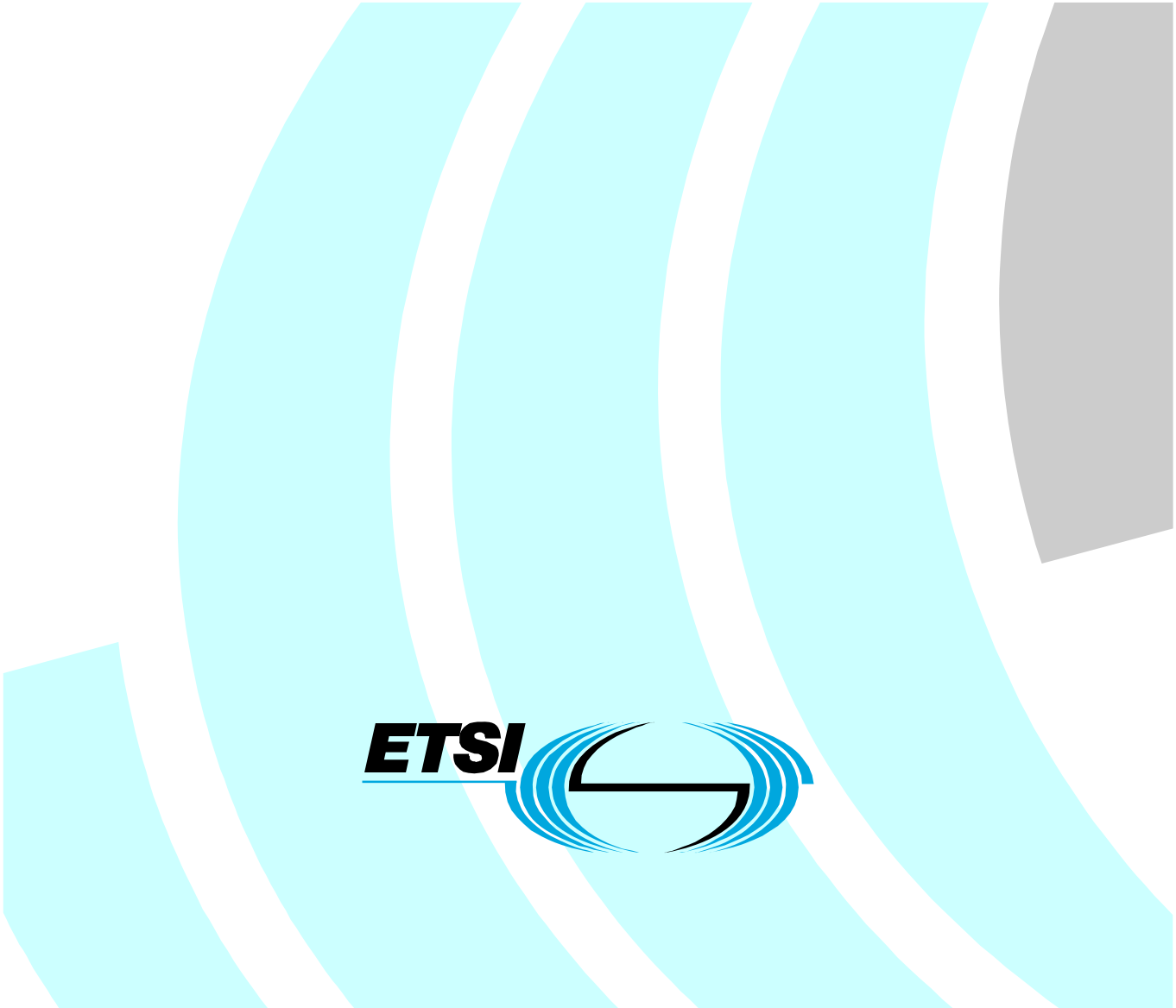
ETSI TR 102 788 V1.1.1 (2010-01)

Technical Report

Methods for Testing and Specification (MTS); Automated Interoperability Testing; Specific Architectures



Reference
DTR/MTS-00116AutoIOP_Arch

Keywords
architecture, intreroperability, methodology, testing

ETSI

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

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

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

Important notice

Individual copies of the present document can be downloaded from:
http://www.etsi.org

The present document may be made available in more than one electronic version or in print. In any case of existing or perceived difference in contents between such versions, the reference version is the Portable Document Format (PDF). In case of dispute, the reference shall be the printing on ETSI printers of the PDF version kept on a specific network drive within ETSI Secretariat.

Users of the present document should be aware that the document may be subject to revision or change of status. Information on the current status of this and other ETSI documents is available at
http://portal.etsi.org/tb/status/status.asp

If you find errors in the present document, please send your comment to one of the following services:
http://portal.etsi.org/chaircor/ETSI_support.asp

Copyright Notification

Contents

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://webapp.etsi.org/IPR/home.asp).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Foreword

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

Introduction

The encouragement of the European Commission for the adoption and promotion of generic test frameworks for the validation of standards based on multiple stacks including middleware provides evidence that successful testing as well as interoperability are key factors to enable the use of new technologies providing all benefits attached to them including competitiveness and innovation. However, technologies are becoming more complex, collaborative and inter-dependant. Therefore, methodologies and approaches for ensuring interoperability need to be innovative and consider new evolving challenges such as the distribution of components and their remote access in an embedded environment. This guide adapts and presents a solid and proven method to these new challenges.

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

Interoperability test suites are usually based on a basic simple idea: key reference points (interfaces) are checked while end-to-end interoperability tests are executed to observe if the message flow conforms to the flows mandated by standards. Each interoperability test suite is compounded by several tests. During interoperability test events, e.g. ETSI Plugtests™, systems developed by different vendors are paired up to execute an agreed set of interoperability tests in test sessions. In such scenarios, automation could help to achieve dramatic time savings. The purpose of testing automation is to reduce time for testing and to avoid repetitive activities which involve a lot of human specialist resources. Automation allows reducing manual interaction related to all the test phases: test execution, trace and message analysis, and reporting. Furthermore, testing automation increases traceability and reliability while it reduces risk of human error. Testing automation helps to assure that the correct evaluation of all the procedures and the parameters foreseen in the testing specification is performed for each test.

The process to automate interoperability testing [i.1] is based on the test description and on the test architecture where system developers should find the unambiguous expected test behaviour, the configuration preconditions, and the network configuration required.

The present document collects example realizations of TTCN-3 based test systems for automated interoperability testing. It provides example applications of the ETSI framework and methodology for automated interoperability testing.

1 Scope

The present document presents the application of the generic framework and methodology for automated interoperability testing. More specifically, it presents its application to the interoperability testing of IP Multimedia Subsystem (IMS) core networks. It includes test architecture, codec and adapter requirements as well as a TTCN-3 test suite that can be used in the context of interoperability events

2 References

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.
- Non-specific reference may be made only to a complete document or a part thereof and only in the following cases:
  - if it is accepted that it will be possible to use all future changes of the referenced document for the purposes of the referring document;
  - for informative references.

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

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

2.1 Normative references

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

Not applicable.

2.2 Informative references

The following referenced documents are not essential to the use of the present document but they assist the user with regard to a particular subject area. For non-specific references, the latest version of the referenced document (including any amendments) applies.

[i.1] ETSI EG 202 810: "Methods for Testing and Specification (MTS); Automated Interoperability Testing; Methodology and Framework".

[i.2] ETSI TS 124 229 (V7.16.0): "Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS); LTE; Internet Protocol (IP) multimedia call control protocol based on Session Initiation Protocol (SIP) and Session Description Protocol (SDP); Stage 3 (3GPP TS 24.229 version 7.16.0 Release 7)".

[i.3] ETSI TS 186 011-2 (V2.3.1): "Technical Committee for IMS Network Testing (INT); IMS NNI Interworking Test Specifications; Part 2: Test description for IMS NNI Interworking".

[i.4] ETSI ES 201 873-5: "Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 5: TTCN-3 Runtime Interface (TRI)".

[i.5] ETSI ES 201 873-6: "Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 6: TTCN-3 Control Interface (TCI)".

[i.6] ETSI ES 201 873-1 (V3.4.1): "Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 1: TTCN-3 Core Language".

[i.7] ETSI ES 201 873-10: "Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 10: TTCN-3 Documentation Comment Specification".

[i.8] ETSI TS 123 228 (V7.15.0): "Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS); LTE; IP Multimedia Subsystem (IMS); Stage 2 (3GPP TS 23.228 version 7.15.0 Release 7)".

[i.9] M. Poikaselkä, G. Mayer, H. Khartabil, A. Niemi: "The IMS: IP Multimedia Concepts and Services", Wiley, 2004.

[i.10] ETSI TS 123 229: "Internet Protocol (IP) multimedia call control protocol based on Session Initiation Protocol (SIP) and Session Description Protocol (SDP); Stage 3".

[i.11] ETSI EG 202 237: "Methods for Testing and Specification (MTS); Internet Protocol Testing (IPT); Generic approach to interoperability testing".

[i.12] ETSI EG 202 568: "Methods for Testing and Specification (MTS); Internet Protocol Testing (IPT); Testing: Methodology and Framework".

[i.13] ETSI EG 186 011-1 (V2.3.1): "Technical Committee for IMS Network Testing (INT); IMS NNI Interworking Test Specifications; Part 1: Test purposes for IMS NNI Interworking".

[i.14] ETSI EG 186 011-2 (V2.3.1): "Technical Committee for IMS Network Testing (INT); IMS NNI Interworking Test Specifications; Part 2: Test descriptions for IMS NNI Interworking".

[i.15] IETF RFC 3261: "SIP: Session Initiation Protocol".

[i.16] IETF RFC 3262: "Reliability of Provisional Responses in Session Initiation Protocol (SIP)".

[i.17] IETF RFC 3265: "Session Initiation Protocol (SIP)-Specific Event Notification".

[i.18] IETF RFC 3313: "Private Session Initiation Protocol (SIP) Extensions for Media Authorization".

[i.19] IETF RFC 3323: "A Privacy Mechanism for the Session Initiation Protocol (SIP)".

[i.20] IETF RFC 3325: "Private Extensions to the Session Initiation Protocol (SIP) for Asserted Identity within Trusted Networks".

[i.21] IETF RFC 3326: "The Reason Header Field for the Session Initiation Protocol (SIP)".

[i.22] IETF RFC 3327: "Session Initiation Protocol (SIP) Extension Header Field for Registering Non-Adjacent Contacts".

[i.23] IETF RFC 3329: "Security Mechanism Agreement for the Session Initiation Protocol (SIP)".

[i.24] IETF RFC 3455: "Private Header (P-Header) Extensions to the Session Initiation Protocol (SIP) for the 3rd-Generation Partnership Project (3GPP)".

[i.25] IETF RFC 3515: "The Session Initiation Protocol (SIP) Refer Method".

[i.26] IETF RFC 3608: "Session Initiation Protocol (SIP) Extension Header Field for Service Route Discovery During Registration".

[i.27] IETF RFC 3841: "Caller Preferences for the Session Initiation Protocol (SIP)".

[i.28] IETF RFC 3891: "The Session Initiation Protocol (SIP) Replaces Header".

[i.29] IETF RFC 3892: "The Session Initiation Protocol (SIP) Referred-By Mechanism".

[i.30] IETF RFC 4028: "Session Timers in the Session Initiation Protocol (SIP)".

[i.31] IETF RFC 4244: "An Extension to the Session Initiation Protocol (SIP) for Request History Info".

[i.32] IETF RFC 5009: "Private Header (P-Header) Extension to the Session Initiation Protocol (SIP) for Authorization of Early Media".

[i.33] IETF RFC 2616: "Hypertext Transfer Protocol -- HTTP/1.1".

[i.34] IETF RFC 2617: "HTTP Authentication: Basic and Digest Access Authentication".

[i.35] IETF RFC 4566: "SDP: Session Description Protocol".

[i.36] IETF RFC 1035: "Domain names - implementation and specification".

[i.37] IETF RFC 2915: "The Naming Authority Pointer (NAPTR) DNS Resource Record".

3 Abbreviations

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

AS	Application Server
ATS	Abstract Test Suite
BGCF	Breakout Gateway Control Function
COPS	Common Open Policy Service
CS	Circuit Switched
CSCF	Call Session Control Function
DNS	Domain Name System
GGSN	3rd Generation Gateway GPRS Support Node
HSS	Home Subscriber Server
IBCF	Interconnection Border Control Function
I-CSCF	Interrogating Call Session Control Function
IETF	Internet Engineering Task Force
IMS	IP Multimedia Subsystem
IP	Internet Protocol
ISC	IMS Service Control
MGCF	Media Gateway Control Function
MGW	Media Gateway Function
MRFC	Multimedia Resource Function Controller
MRFP	Multimedia Resource Function Processor
MTC	Main Test Component
NNI	Network-to-Network Interface
P-CSCF	Proxy Call Session Control Function
PDF	Policy Decision Function
PSTN	Public Switched Telephone Network
PTC	Parallel Test Component
QoS	Quality of Service
S-CSCF	Serving - Call Session Control Function
SEG	Security Gateway
SGSN	3rd Generation Serving GPRS Support Node
SGW	Signalling Gateway
SIP	Session Initiated Protocol
SLF	Subscription Locator Function
SUT	System Under Test
TCI	TTCN-3 Control Interface
TD	Test Description
THIG	Topology Hiding Inter-networkGateway
TrGW	Transition Gateway
TSI	Test System Interface
UE	User Equipment
VoIP	Voice over Internet Protocol
XML	eXtended Mark up Language
XSD	XML Schema Definition

4 Interoperability Testing of IP Multimedia Subsystem core networks

The following clause introduces briefly IP Multimedia Subsystem (IMS) technology basics. For more detailed information the reader is referred to [i.8] and [i.9].

4.1 IP Multimedia Subsystem

With the exploded usage of the Internet and the distribution of more and more attractive terminals with enhanced multimedia features such as colour and high definition displays, embedded cameras, applications like media readers, games and the global positioning system, a need to provide users with the capability to share these contents arose. Therefore, the usage of a terminal needed to be extended by video and media content beyond voice call. These requirements led to the introduction of a Internet Protocol (IP) based peer-to-peer architecture, the IP Multimedia Subsystem (IMS). IMS is mainly based on the use of Session Initiation protocol (SIP) enabling clients to invite other clients to a session and negotiate control information about the media channels needed for the session. In addition, IMS provides control capabilities (e.g. authentication of clients), architecture capabilities (e.g. network to network interfaces) and administration capabilities (e.g. charging) to the network operators.

With IMS and new "all-in-one terminals", users can enter to a wide set of intelligent, interactive, location based, and multimedia services. This includes television, file sharing, instant messaging, chat, presence, e-mail, group management, and conferencing. As depicted in figure 1, IMS is an architecture for the convergence of data and speech communications and for the convergence of networks. IMS services and applications can be implemented independent of different access networks such as mobile, fixed, broadband, and corporate networks.

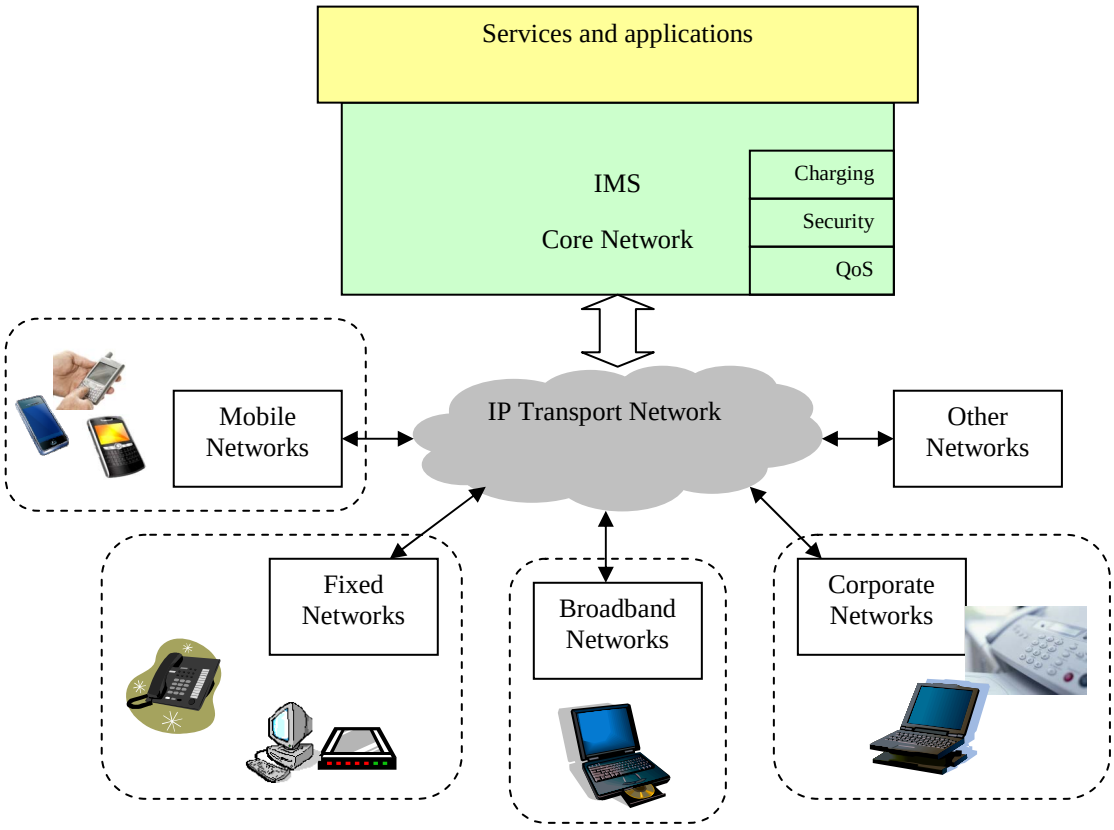


Figure 1: IMS and its access networks

4.2 IMS architecture

The clause describes the IMS architecture including its reference points. As depicted in figure 2, the IMS network architecture constitutes the following three layers:

- the user plane;
- the control plane; and
- the application plane.

This logical splitting of IMS functionalities facilitates the addition of new access networks and makes services independent from the access network.

IMS network entities can be distinguished into the following functional categories:

- support:
  - PDF: the Policy Decision Function verifies policy to the requested IP flows, returning this information to the S-CSCF serving the User Equipment (UE);
  - SEG: the Security Gateway protects control plane traffic between security domains (i.e. network operator domain);
  - THIG: the Topology Hiding Inter-network Gateway hides the configuration capacity an topology of the network from outside;
- services:
  - AS: Application Server tasks include the processing of an incoming SIP session, originating SIP requests, sending account information to charging function;
  - MRFC and MRFP: Media Resource Function Controller and Processor provide the needed mechanism for bearer related services on media streams (e.g. mixing in conferencing service);
- registration, session management and routing including:
  - P-CSCF: the Proxy Call Session Control Function (CSCF) is the entering point in the IMS network and performs SIP compression, IPSec security tasks, interaction with PDF and emergency session detection;
  - I-CSCF: the Interrogating CSCF is the contact point for all the connections for the involved network operator subscriber. I-CSCF identifies next hop (S-CSCF or AS) from HSS, assigns S-CSCF, routes incoming requests to the assigned S-CSCF or AS, provides THIG functionality. The IBCF can be considered part of the I-CSCF because it acts as an I-CSCF when the session involves another interconnected IMS network and border control concepts are applied;
  - S-CSCF: the Serving CSCF is the core of the IMS control network and it is involved in handling registration, making routing decision, keeping session states, storing service profile;
- databases:
  - HSS: the Home Subscriber Server stores user identities (private and public), Registration information, access parameters and service-triggering information;
  - SLF: the Subscription Locator Function implements a resolution mechanism for I-CSCF, S-CSCF and AS to resolve the HSS address where the user data are stored for a given subscriber;
- interworking: this functionality allows to interconnect IMS user and services with network in another domain (e.g. CS network). When there is the need to break out to another network domain, The S-CSCF sends the SIP session request to the Breakout Gateway Control Function (BGCF) which interoperate to the other network using the Signalling Gateway (SGW) for signalling and IMS Media Gateway Function (IMS-MGW) for the user plane. The BGCF controls the IMS-MGW with the Media Gateway Control Function (MGCF);
- charging.

In figure 2, the IMS architecture is qualitative described in a layered layout, highlighting with grey shadows the functionalities described above. This figure is not exhaustive, e.g. charging functions, SEG, THIG are missing.

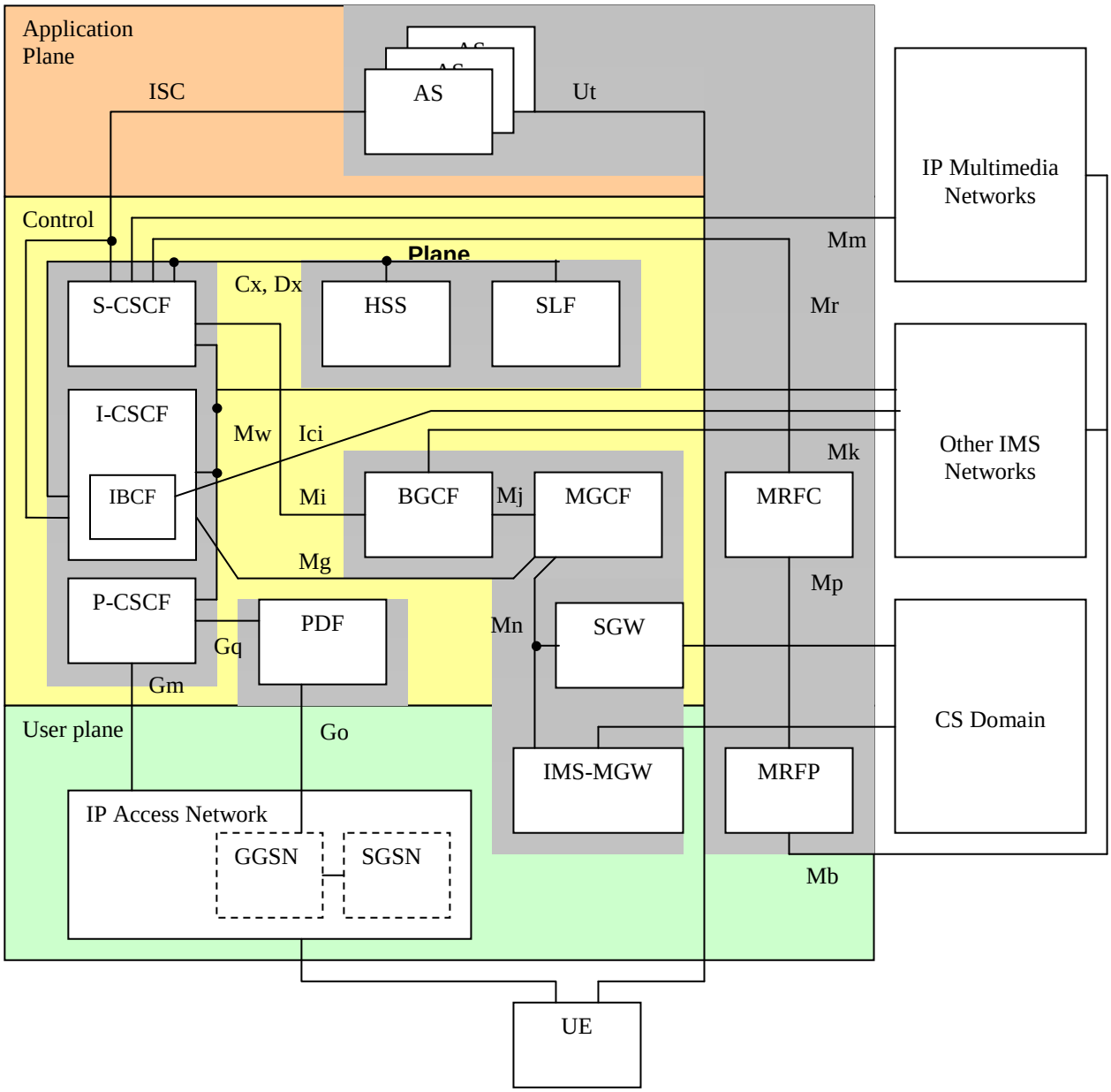


Figure 2: IMS network architecture

The network entities are connected by IMS reference points. Referring to figure 2, it is possible to derive the main reference points. These are described in table 1.

Table 1: IMS main reference Points

Name	Connected Entities		Description	Protocol
ISC	S-CSCF I-CSCF	AS	An AS hosts and executes services. ISC routes either the initial SIP request to the AS or an AS-initiated SIP request.	SIP
Ut	UE	AS	The Ut provides users with functionalities to manage and configure their services.	HTTP
Mm	S-CSCF I-CSCF	IP Multimedia Network	Mm is introduced to communicate with other IP Multimedia Networks.	SIP
Cx	S-CSCF I-CSCF	HSS	The Cx procedures belong to location management, user data handling and user authentication categories.	Diameter
Dx	S-CSCF I-CSCF	SLF	The Dx permits to deal with HSS resolution procedure in multiple HSS configuration.	Diameter
Mw	P-CSCF S-CSCF I-CSCF	P-CSCF S-CSCF I-CSCF	The Mw connects the different CSCFs, handling registration, session control and transaction procedures.	SIP
Mk	BGCF	BGCF of other IM core network Subsystems	The Mk forwards the session to the other network BGCF when a breakout occurs involving another network.	SIP
Mj	BGCF	MGCF	The Mj forwards the session to the MGCF when a breakout occurs in the same network.	SIP
Mi	S-CSCF	BGCF	The Mi enables to forward a session which needs to be routed to the CS domain from the S-CSCF to the BGCF.	SIP
Mg	MGCF	I-CSCF	The Mg connects the MGCF to IMS, forwarding an incoming session signalling from the CS domain to the I-CSCF.	SIP
Mp	MRFC	MRFP	Used by the MRFC to control the MRFP.	H.248
Mn	MGCF	IMS-MGW	The Mn controls the user plane and its main tasks are reserve and connect terminations, connect or release echo cancellers to termination, connect or release tones and announcements to terminations, send/receive DTMF tones.	H.248
Mr	S-CSCF	MRFC	The Mr provides a mean of communication for the S-CSCF to the MRFC when bearer related service are needed	SIP
Gm	UE	P-CSCF	The Gm connects the UE to the IMS, handling registration, session control and transaction procedures.	SIP
Go	PDF	GGSN	The Go procedures can be divided in media authorization (QoS assurance and control policies) and charging correlation between IMS and GPRS.	COPS
Gq	P-CSCF	PDF	The Gq is used to exchange or to setup policy information with the PDF.	Diameter
Mb		IP network		
Ici	IBCF	IBCF of other IM core network Subsystems	The Ici allows IBCFs in different IM CN Subsystem to communicate in order to provide the communication and forwarding of SIP messages.	SIP

4.3 Interoperability testing of IMS core networks

At ETSI, the Technical Committee for IMS Network Testing (TC INT) works on the specification of tests for the assessment of different aspects of IMS core networks and their compliance to the IMS standards. One of its test specifications [i.13], [i.14] describes the use of interoperability testing to check conformance of IMS core network elements to TS 124 229 [i.2]. The specification splits into two parts: (conformance) test purposes and interoperability test descriptions.

The IMS interoperability test specification assesses the interworking of two IMS core network implementations at their network to network interface (NNI) in different configurations. It assesses interoperability and conformance at the NNI in conditions such as IMS interworking, IMS roaming, and topology hiding as well as with the integration of 3rd party application servers via the ISC interface. Finally, it also addresses IMS interworking with legacy Public Switched Telephone Network (PSTN) systems.

Note that IMS core networks are viewed by this test specification as a black box, i.e. a physical entity that cannot be separated in any way. All the functions of an IMS network may, but do not have to be implemented in one system. When entities are co-located standardized interfaces turn into product internal interfaces (and may therefore not accessible for protocol monitors) and leaves only Gm, ISC, Mw, and Ic reference points for analyzing communication for standard compliance.

The automated testing architecture presented in the following clauses realizes the tests described in [i.14] and has been developed based on the concepts and processes presented in the ETSI framework and methodology for automated interoperability testing of distributed systems [i.1].

5 Abstract test suite specification

This clause describes the Abstract Test Suite (ATS) specification used for IMS core network NNI interoperability testing.

5.1 Test configuration

Test configurations have been defined in [i.14] by applying an interface based design approach. Here, each monitored IMS interface is paired with one dedicated Parallel Test Component (PTC) which receives all relevant message information from the TTCN-3 SUT Adapter (SA) via the abstract test system interface and checks its correctness according to the conformance criteria listed for a particular IMS test. An example test configuration is shown in figure 1. For detailed discussion of the abstract test system interface, the reader is referred to clauses 6.2 and 6.3 of the present document.

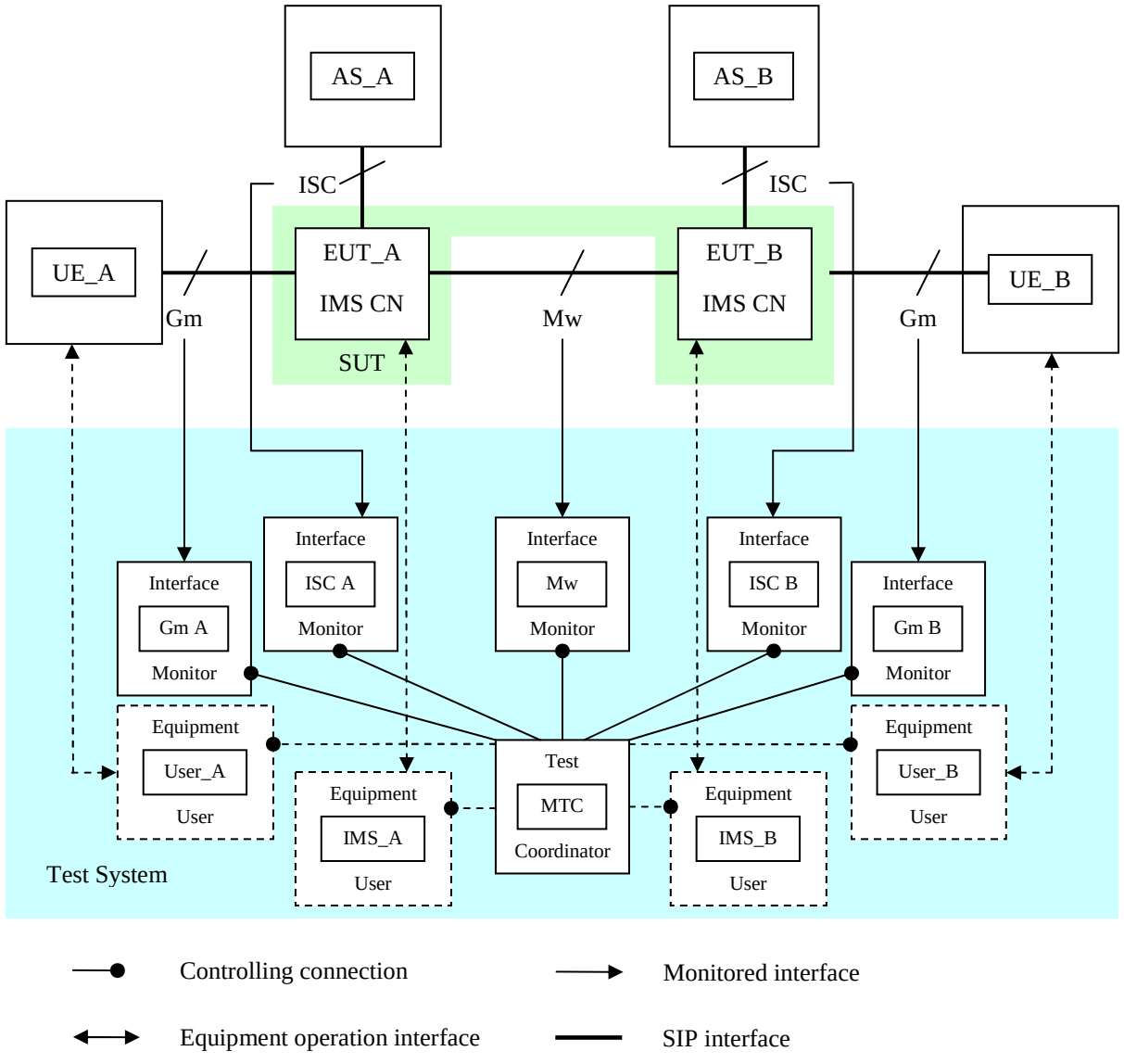


Figure 3: Example IMS NNI interoperability test system configuration

The test system configuration is based on the general TTCN-3 test system architecture specified in ES 201 873-5 [i.4] and ES 201 873-6 [i.5] as well as on the concepts stated in EG 202 810 [i.1]. Note that figure 3 does not illustrate roaming and IMS/PSTN interworking aspects in a test configuration. In addition, it does not show the DNS server as an application support node for each IMS core network as well as its associated interface monitor component, which are only required in a few tests. Note that TTCN-3 test components which are shown with dashed lines in figure 3 are only started for the execution of the test suite in live mode.

The different types of TTCN-3 components used in this ATS are:

- **Test Coordinator** is a component type is dedicated to coordinate the behaviour of all other test components, which work on tasks independently of each other. It is in charge of controlling the overall execution, management of testing phases, conformance verdict and end-to-end interoperability verdict management, and synchronization.
- **Equipment User** is a component type is dedicated to handle equipment operation, e.g. configure an IMS CN, make basic call or messaging from a UE, check for an incoming call notification on a UE, barring a user in a IMS CN, de-register a user forcefully from the IMS CN during a call.
- **Interface Monitor** is a component type that is dedicated to monitor one specific logical interface either between two EUTs or a EUT and an Application Support Node, e.g. IMS CN and a DNS server.

5.2 Test design guidelines

This clause defines guidelines and design patterns used in the Abstract Test Suite (ATS).

The ATS is specified using TTCN-3 [i.6]. The benefits of TTCN-3 are:

- well defined syntax;
- well defined static and operational semantics;
- rich type system which includes concepts like a verdict & native list types, subtyping, type compatibility, etc.;
- powerful built-in matching mechanism and matching expressions;
- snapshot semantics:
  - ensures and preserves order of external event arrival;
  - allows checking of each external event against a number of alternative constraints;
- allows definition of concurrent tests, i.e. tests with multiple test components;
- support for asynchronous as well as synchronous communication paradigms;
- support for dynamic test configurations, i.e. that test components can be (re)mapped, (re)connected or (re)created on the fly during a test;
- allows specification of execution parameters at run time via module parameters to ease adaptation of test suite to different testing environments;
- support for timers;
- enables completely automated test execution.

5.2.1 TTCN-3 naming convention

TTCN-3 can be considered a programming language. Therefore, the usage of naming conventions supports or increases code readability, consistency, and maintainability of the code. It also helps to achieve earlier detection of semantic errors and the distribution of test suite development work across several developers.

The naming convention used by this test suite is based on the ETSI generic naming conventions and follows the underlying principles:

- when constructing meaningful identifiers, the general guidelines specified for naming in clause 8 of EG 202 568 [i.12] should be followed;
- the names of TTCN-3 objects being associated with standardized data types (e.g. in the base protocols) should reflect the names of these data types as close as possible (of course not conflicting with syntactical requirements or other conventions being explicitly stated);
- the subfield names of TTCN-3 objects being associated with standardized data type should also be similar to corresponding element names in the base standards (be recognizable in the local context);
- in most other cases, identifiers should be prefixed with a short alphabetic string (specified in table 2) indicating the type of TTCN-3 element it represents;
- prefixes should be separated from the body of the identifier with an underscore ("_");
- only test case names, module names, data type names and module parameters should begin with an upper-case letter. All other names (i.e. the part of the identifier following the prefix) should begin with a lower-case letter.

Table 2 specifies the naming guidelines for each construct of the TTCN-3 language indicating the recommended prefix and capitalization.

Table 2: Naming Conventions

Language element	Naming convention	Prefix	Example	Notes
Module	Upper-case initial letter	none	LibSip_TypesAndValues	
Group	Lower-case initial letter	none	messageGroup	
Data type	Upper-case initial letter	none	SetupContents	
Message template	Lower-case initial letter	m_	m_response	See note 1
Message template with wildcard or matching expression	Lower-case initial letter	mw_	mw_response	See note 2
Modifying message template	Lower-case initial letter	md_	md_response	See note 1
Modifying message template with wildcard or matching expression	Lower-case initial letter	mdw_	mdw_reponse	See note 2
Port instance	Lower-case initial letter	none	configPort	
Test component reference	Lower-case initial letter	none	userTerminal	
Constant	Lower-case initial letter	c_	c_maxRetransmission	
Constant (defined within component type)	Lower-case initial letter	cc_	cc_maxRetransmission	
External constant	Lower-case initial letter	cx_	cx_macId	
Function	Lower-case initial letter	f_	f_authentication()	
External function	Lower-case initial letter	fx_	fx_calculateLength()	
Altstep (incl. Default)	Lower-case initial letter	a_	a_receiveSetup()	
Test case	All upper-case letters	TC_	TC_IMS_MESS_0001	
Variable (defined locally)	Lower-case initial letter	v_	v_macId	See note 3
Variable (defined within component type)	Lower-case initial letter	vc_	vc_systemName	
Timer (defined locally)	Lower-case initial letter	t_	t_wait	
Timer (defined within component type)	Lower-case initial letter	tc_	tc_authMin	

Language element	Naming convention	Prefix	Example	Notes
Module parameter	All upper-case letters	none	PX_MAC_ID	
Parameterization	Lower-case initial letter	p_	p_macId	
Enumerated Value	Lower-case initial letter	e_	e_syncOk	
NOTE 1: This prefix should be used for all template definitions which do *not* assign or refer to templates with wildcards or matching expressions, e.g. templates specifying a constant value, parameterized templates without matching expressions, etc. NOTE 2: This prefix should be used in identifiers for templates which either assign a wildcard or matching expression (e.g. ?, *, value list, ifpresent, pattern, etc) or reference another template which assigns a wildcard or matching expression. NOTE 3: In this case it is acceptable to use underscore within an identifier.				

NOTE: Naming conventions have been enforced only in the TTCN-3 code written within this project for this ATS. There may be some minor deviations from these conventions in code that has been reused from other ETSI projects.

In addition to the above naming conventions, TTCN-3 functions which specify behaviour that is to execute on the main test component should use a "f_mtc_" prefix to distinguish it from functions which can run on PTCs which have no prefix extension. For further information on function design the reader is referred to clause 5.2.4.

5.2.2 TTCN-3 language version

This test suite has been developed based on the concepts available in version 4.1.2 of the TTCN-3 core language. In order to simplify codec and test implementation, this test suite avoids and should avoid in future versions the use of nested TTCN-3 type definitions as well as features deprecated in this version of the language, e.g. the use of the all keyword in TTCN-3 port type definitions, or port types of type mixed.

5.2.3 Modularization

The ATS has been specified by using a library approach for TTCN-3 modules. Here, reusable definitions have been isolated from ATS specific definitions in so called "TTCN-3 Libraries". TTCN-3 libraries are specified as source code since the TTCN-3 standards do not define the integration of pre-compiled libraries. ATS and library specific modules are distinguished in their prefix which is either "Ats" or "Lib".

The following prefixes are used in module names to identify ATS specific and library TTCN-3 modules:

- **LibCommon:** a collection of generally useful TTCN-3 definitions for any test suite implementation, e.g. basic types definitions, verdict handling, timing, test component synchronization.
- **LibUpperTester:** a collection of reusable TTCN-3 definitions related to upper tester specification for conformance and/or interoperability testing including an abstract equipment operation protocol.
- **LibSip:** a collection of reusable TTCN-3 definitions related to SIP standards including type definitions for SIP base RFC as well as other RFCs, dummy, base and specific SIP templates.
- **LibIms:** a collection of reusable TTCN-3 definitions related to IMS specific definitions including test component state information.
- **LibIot:** a collection of reusable TTCN-3 definition for any IOT test suite implementation aligned with the ETSI methodology for automated interoperability testing of distributed systems.
- **AtsImsIot:** IMS NNI IOT specific TTCN-3 definitions, e.g. test configuration management, test case statements, and test purpose checking functions.

In general, TTCN-3 libraries contain either following types of modules or types of groups within a module:

- **TypesAndValues** collects library specific TTCN-3 type and constant definitions.
- **PIXIT** collects module parameter declarations used by library definitions.
- **TestInterface** contains component and port type definitions reflecting the interface(s) handled by the library.
- **Templates** collects library specific TTCN-3 template definitions, e.g. its Behavior module(s).

- **Functions** collects generic TTCN-3 and external functions.
- **Behavior** collects generic TTCN-3 functions expressing elementary message exchanges.

For more information regarding the specific TTCN-3 libraries used by this ATS the reader is referred to clause 5.3. The ATS specific part of the test suite contains the following types of modules:

- **PICS** collects test case selection module parameters associated with the ATS.
- **PIXIT** collects module parameter declarations used by ATS definitions.
- **TypesAndValues** collects ATS specific TTCN-3 type and value definitions except component and port types.
- **TestSystem** specifies TTCN-3 component type definitions used by to create MTC and PTCs in the test cases as well as the abstract test system component type, i.e. the system component type. This module either specifies component types based on port types (respecting component type compatibility) or by extending component types defined in one or more TTCN-3 library interface modules. Component types may also add ATS specific variables or ports.
- **Templates** collects ATS specific TTCN-3 template definitions, e.g. used by its behaviour module(s).
- **TestConfiguration** contains functions which realize the configuration of the test system, i.e. the mapping of test components for the establishment and tear down of different test configurations as well as the configuration of the SUT Adapter.
- **Functions** collects ATS specific TTCN-3 functions.
- **Behavior** collects ATS specific TTCN-3 functions for checking conformance related to test purposes associated with test descriptions.
- **TestControl** contains the control part definition which performs test case selection.
- **TestCases** collects TTCN-3 test case definitions which should be split across multiple modules of this type, e.g. for grouping test case according to functionalities.

5.2.4 SIP message template design

IMS SIP templates are defined in the IMS and SIP libraries using a three step approach:

- In the first step, for every message type and direction (sending or receiving) a dummy template is defined, e.g. m_ACK_Dummy and mw_ACK_Dummy. All optional fields of the dummy template are either set to 'omit' or '*' depending on the direction. Mandatory fields are set to dummy values or '?'. Please note that dummy templates should never be used directly for sending or receiving!
- In the second step, base templates are derived from the dummy templates. These base templates set all main SIP headers to specific (parameterized) values which are in accordance to the SIP standard. The template identifiers of these modifications include the keyword "base", e.g. md_ACK_Request_Base or mdw_ACK_Request_Base.
- In the third step, any other templates, e.g. templates for setting IMS specific headers are derived from the base templates by modifying the fields that need to be restricted for specific purposes, e.g. md_ACK_Request_route etc. These specific templates should be mainly used for sending and receiving.

This design approach allows the extension of SIP message types with additional headers with a minimal change in templates, i.e. the dummy templates. All other templates do not require updates. The adoption of this design approach in new template additions to the ATS will help to improve the maintainability and continue the readability of the test suite.

5.2.5 Function design

The approach selected for the design of functions maximizes reuse as well as clearly separates and isolates behaviour specific to the ATS, IOT, SIP, or IMS to their respective libraries.

Test case statements are defined in the ATS by following the naming of IMS NNI test descriptions as closely as possible, i.e. it invokes a configuration function named after the IMS NNI test configuration, then a preamble function representing the initial conditions of the TD, followed by equipment operation functions resembling the test sequence interleaved with test purpose checking functions. All of the functions called from the test case statement are functions which run on the main test component (MTC).

Test configuration functions are defined in AtsImsIot and create all required test components, mappings and connections, as well as configuration. For example, these functions encapsulate the setting of filters in the adapter for interface monitor components.

However, test sequence functions are named after the task they perform on a parallel test component (PTC). Therefore, the additional prefix "mtc_" has been introduced to clarify that even these functions run on the MTC. The main purpose of these functions is to start the behaviour specified in the function identifier on a specified PTC.

Equipment operation functions which execute on PTCs are defined in LibIms behavioural modules. These customize the generic equipment operation function defined in LibUpperTester with the commands defined in the LibIms. The functions set the PTC E2E verdict by calling the verdict handling mechanism offered by LibIot.

Test purpose checking functions are defined in AtsImsIot. These functions are split into two separate functions since each test purpose requires checks on two separate logical interfaces. Test purpose functions are implemented based on a generic function `f_imsIot_receive()` which is parameterized, e.g. with the templates performing part of the checks specified in a specific TP. In addition, this function allows passing received messages automatically to the MTC where it can be checked if specific content is equal to the one in other messages. `f_imsIot_receive()` is specified based on the generic `f_gen_receive()` function implemented in LibIot. The PTC conformance verdict is set by calling the verdict handling mechanism offered by LibIot from both the IMS IOT specific and the general receive function.

5.2.6 Test case orchestration

This IMS IOT test suite has been specified using multiple test components. Each of them handles different and independent tasks. Specific test components are only required in certain phases of a test, e.g. equipment operation components for configuration during the preamble, interface monitors during the execution of the test body.

When specifying a TTCN-3 test with multiple test components, it is possible to either run PTCs in parallel or to start and stop their behaviour sequentially from the test coordinator component. In the parallel approach, test components have to be synchronized explicitly via test component messaging, e.g. by using the synchronization functions provided in LibCommon. In the sequential approach, alive components are used which allow restarting different behaviour functions on component instances.

EXAMPLE 1: When a sequential execution approach is used to trigger UE_A to send a SIP request to UE_B, the test is realized by first starting the equipment user test component User_A to request to operate UE_A to generate the SIP request and wait until a return from the equipment operation request; then to start the interface monitor component Gm_A to check if the correct SIP request as been observed and to wait until it returns from this check; then to start the interface monitor component Mw to check if the expected SIP request is observed and to wait until it returns from this check.

EXAMPLE 2: In the concurrent approach, all components are started after each other without waiting for the completion of other components; then they have to be synchronized, either after the message flow is completed or all checks relevant to the test have been performed; finally they are triggered to continue their execution and synchronized again.

Since test cases specification based on the sequential approach resembles more of a direct encoding the test sequences and call flows of the IMS test descriptions, it has been selected as the approach to manage the execution of test cases with multiple test components in this ATS.

5.2.7 Handling of proprietary interfaces

Equipment user test components (see figure 1) have the purpose to configure or trigger IMS equipment to perform particular tasks like barring a user from an IMS CN, to register to services from an UE or to initiate a call from a UE. The actual user interface of IMS equipment and its operation are however not standardized and highly implementation dependent. However, some manufacturers provide special automatable interfaces for the purposes of testing, e.g. AT based command set for IMS UEs.

The TTCN-3 ATS should be implemented agnostic of proprietary interfaces. It uses an abstract concept for equipment operations which are based on a command request and response. Commands are abstract descriptions of actions to be taken, e.g. enter contact or initiate a VoIP call. Abstract primitives may have abstract parameters, e.g. the terminating user identifier.

It is assumed that a part or component of the TTCN-3 SUT Adapter (not shown in figure 1) provides a mapping or translation of the abstract TTCN-3 equipment operation requests sent by equipment user test components to actual IMS equipment, specific operations or a terminal like output device that instructs test equipment operators in their interaction with the IMS equipment. Responses or observations of IMS equipment responses have to be mapped in the SUT Adapter to abstract responses prior to being sent to the respective equipment user test component. This mapping from abstract to concrete operations for the equipment participating in a test is beyond the scope of the TTCN-3 ATS and needs to be addressed as part of the TTCN-3 SUT adapter implementation. Note that one test component should be implemented for each IMS-equipment that is planned to be used during testing.

EXAMPLE: The equipment user test component User_A may initiate a call from UE A by sending an InitiateVoIPCallReq(DestUserInfo) command via the TSI to the SUT adapter. This command can be translated into a UE operation instruction which is displayed to the operator of UE A via a terminal window.

5.2.8 Message skipping

Analysing the messages exchanged between two or more EUTs by an interface monitor test component can be complex when the messages to be checked are part of a longer message exchange on the monitored interface. In addition, IOT traffic captures may contain messages sent as part of the preamble or other unanticipated traffic that offsets the message observation from its anticipated occurrence in the test description call flow. For such cases, it is first necessary to locate the beginning of the sequence to be analysed, i.e. to skip all the preceding messages. This issue is unique to interoperability testing and not trivial to solve.

In the case of IMS NNI testing, a test may assume that the all the UEs are already registered at the beginning of the test. However, in the test execution the UEs may not be registered at the beginning of the test, they first have to initiate a SIP register request before any other actions. Similarly, a test execution trace gained with manual triggering of EUTs may include unsuccessful attempts to run the same test due to configuration problems like mistyping of a SIP URI. It has been decided to address such message skipping by providing a time stamp to the test adapter from which it should start parsing for relevant messages.

Another issue that has been addressed is the need to skip messages that appear in the call flow but which do not have to be checked from the test description point of view. This has been implemented as a part of the AtsImsIot `f_imsIot_receive()` function for interface monitor components and allows skipping of a number of any messages (of that protocol) or skipping a number of specific type of messages, e.g. SIP INVITE messages.

5.2.9 Management of EUT interface information

This clause describes two generic approaches to provide IMS core network interface information to a TTCN-3 test system, e.g. user, domain, and IP address information. Since the configuration of a test suite with EUT interface information can be a laborious and tedious task, and may even change between different interoperability testing sessions, a management system for such information is needed as part of a test system to handle updates on this information quickly and without introducing errors. This system needs to fulfil the following requirements:

- Generic, i.e. is should be usable for different kind interfaces of EUTs.
- Compilation-independent: If EUT interface information changes, it should not be required to rebuild or recompile a test system in order to update the tests with the latest information.

In the following paragraph, two approaches are described. The module parameter based approach has been implemented in the ATS. However, the XML approach still remains an option in further ATS evolutions.

5.2.9.1 Module Parameter Approach

In this approach, EUT interface information is entered within TTCN-3 data structure using TTCN-3 module parameters. Here, a well defined TTCN-3 data structure for the specification of the EUT interface information is stringly required since the types are reflected in module parameter value specifications. The type should improve understandability and readability of module parameter values. In addition, a number of functions are needed to extract the product specific EUT information, e.g. for checking of correct message values from the data structure.

TTCN-3 module parameters can also be changed after compilation. However, the syntax of module parameter value provision is not standardized and varies between different TTCN-3 tools. In addition, IMS equipment vendors are not likely to be able to provide their equipment information in this syntax. Therefore, it is expected that equipment information data is entered by the test engineer executing the TTCN-3 test cases.

5.2.9.2 XML Approach

Figure 4 describes an XML based approach for the management of the EUT interface information. The approach is in many aspects similar to the previous one. The difference is that instead of module parameters an XML file is used to specify equipment information. A well designed TTCN-3 data structure and functions for extracting specific product information are also required as in the first approach.

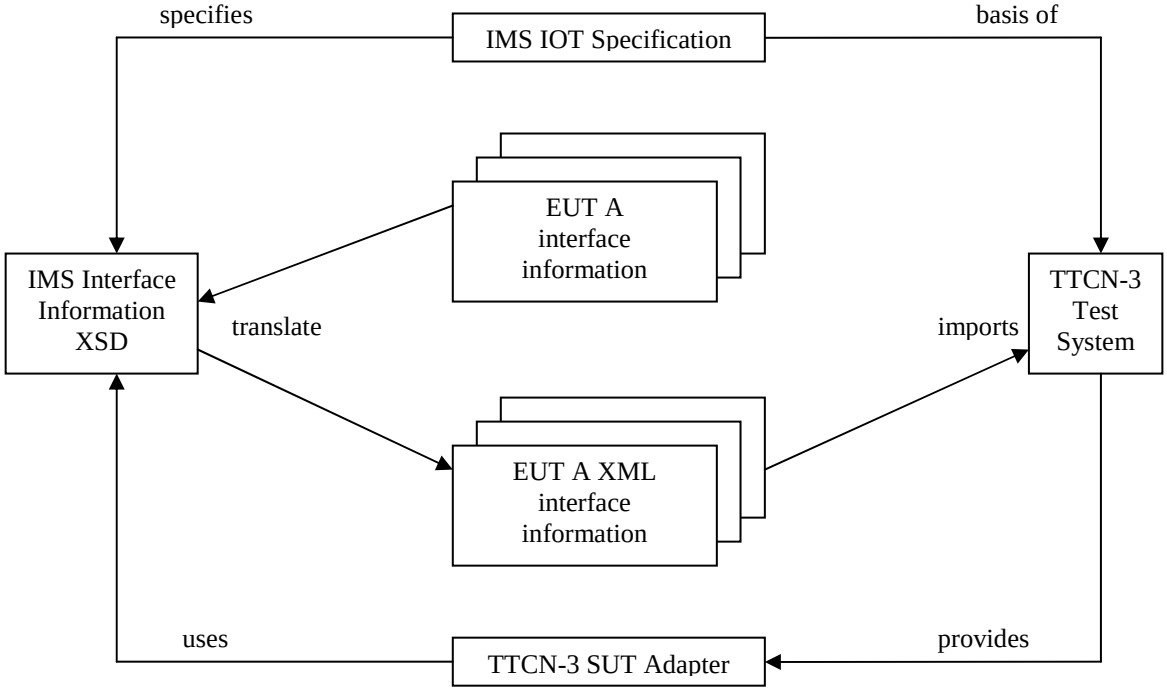


Figure 4

In this approach, it is possible to include in addition the to test descriptionsan XML Schema Document (XSD), which specifies interface information data structures for different types of EUTs or equipment based on the TTCN-3 equipment information data structure, in the IMS IOT specification.

During test execution, the TTCN-3 test system is then informed about the location of the vendor specific XML document location via PIXIT (module) parameters. XML based equipment information for each EUT interface data is then copied to the TTCN-3 equipment information data structure by using the TTCN-3 XML mapping. The XML information can also be provided to the TTCN-3 SUT test adapter which can automatically extract configuration information via the XSD.

This approach is not IMS specific and is virtually applicable for any interoperability test suite. Another advantage of this approach is that XML is standardized and an XSD file can be used for validating the address information data. Participants to interoperability events could fill out and validate the predefined XML scheme prior to providing interface information about their products. The drawback of this approach is that it requires the ability of TTCN-3 tools to allow import of XML or otherwise the implementation and integration of a complex external function.

5.2.10 Documentation

In order to allow browsing of the ATS without a TTCN-3 editor, the test suite has been documented using standardized TTCN-3 documentation tags [i.7]. These tags can be extracted and turned into HTML based documentation. The main documentation tags used in the documentation of this ATS are summarized in table 3.

Table 3: Used TTCN-3 Documentation Tags

Tag	Description
@author	Specifies the names of the authors or an authoring organization which either has created or is maintaining a particular piece of TTCN-3 code.
@desc	Describes the purpose of a particular piece of TTCN-3 code. The description should be concise yet informative and describe the function and use of the construct.
@remark	Adds extra information, such as the highlighting of a particular feature or aspect not covered in the description.
@see	Refers to other TTCN-3 definitions in the same or another module.
@url	Associates references to external files or web pages with a particular piece of TTCN-3 code, e.g. a protocol specification or standard.
@return	Provides additional information on the value returned by a given function.
@member	Documents a member of structured TTCN-3 definitions.
@param	Documents a parameter of parameterized TTCN-3 definitions.
@version	States the version of a particular piece of TTCN-3 code.

The following provides some basic guidelines on the usage of tags for specific TTCN-3 definitions:

- each TTCN-3 module should use the *@author*, *@version* and *@desc* tags;
- the *@desc* tag should be used with all TTCN-3 definitions. However, this should not be taken to the extreme. For example, it is probably not useful to tag literally every single constant or template declaration. It is left to the discretion of the writer to find the right level of use. At least all major constructs such as test cases and functions should have a comprehensive description:
  - when a TTCN-3 definition uses module parameters, it is also recommended to mention this explicitly in the description;
  - descriptions for behavioural constructs should mention if they set the test component verdict and also all known limitations of the construct;
  - descriptions for type definitions, e.g. component types, should mention if the type has been designed to be type compatible to another type or vice versa to be used as a basis for other type definitions;
- the @see tag should be used to make dependencies between TTCN-3 definitions which are described by a @desc tag more explicit in the documentation, e.g. if some TTCN-3 definition uses a module parameter then its TTCN-3 definition should be referenced to using a @see tag;
- where applicable, parameterized constructions such as functions, altsteps and templates should use the *@param* and *@return* tags. The @param tags should first list the parameter name and then a brief description of how this parameter is used by the construct;

- the *@url* tag should be used to refer to the specification from which the TTCN-3 definition was derived from, e.g. a type definition could refer to a particular RFC IETF page. In some cases it may be necessary to use the *@desc* tag instead for this purpose as documents often are hard to access internally, i.e. it may only be possible to specify a reference to a complete document but impossible to point to a very specific clause in the present document;
- the *@url* tag may be used to link to relevant documentation such as Test Purposes or original requirements or even drawings of test configurations. Generally, the corresponding Test Purpose (in the TSS&TP) and to the corresponding Requirement (in the Requirements Catalogue) should be linked from the relevant TTCN-3 test case definition;
- the *@remark* tag may be used with any TTCN-3 definition. It should be used sparingly, e.g. possibly to indicate how a TTCN-3 definition should not be used.

5.3 Mapping of test descriptions to test cases

The ATS define one test case (TC) per IMS NNI test description (TD).

The following naming convention is used by the ATS for test cases:

Test case name	**=**	**<TC_PREFIX>_<TD_ID>**	
<TC_PREFIX>	=	the test cases prefix as specified	e.g. "TC_"
		in the TTCN-3 naming conventions	
<TD_ID>	=	the test description Id	e.g. "IMS_MESS_0001"

6 Test system aspects

TTCN-3 codecs and test adapters that are used with this ATS should conform to re requirements defined in this clause.

6.1 Test system architecture

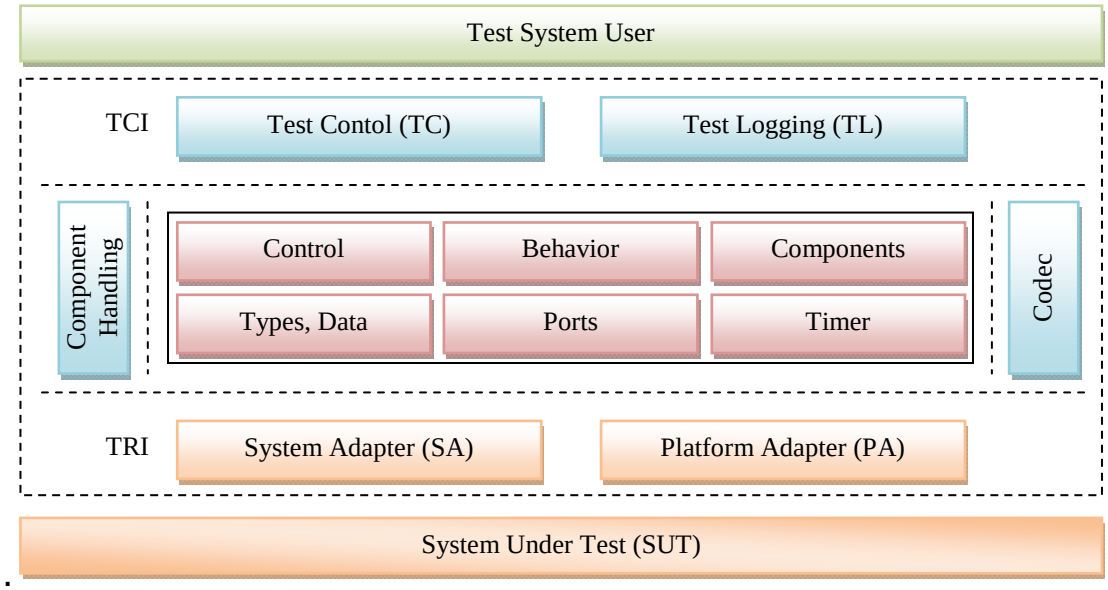


Figure 5: Abstract Test System Architecture

Figure 5 shows the abstract test system architecture. It shows a test system (TS) that is compliant to [i.4] and [i.5] supporting two interfaces: the TTCN-3 Control Interface (TCI) handling the interaction between the TTCN-3 Executable (TE) and the Test Management (Test Control and Test Logging), and the TTCN-3 Runtime Interface (TRI) which handle the communication between the TE and System/Platform Adapter. For further descriptions, the reader is referred to [i.4] and [i.5].

6.2 SUT adapter requirements

Figure 6 illustrates the port association between the TTCN-3 executable (TE) and System Adapter (SA). The SA includes the lower and the upper tester. The lower tester implements the monitoring of the real interface and the upper tester translates the abstract equipment operation request into a concrete one or interacts via instructions with the user of that equipment.

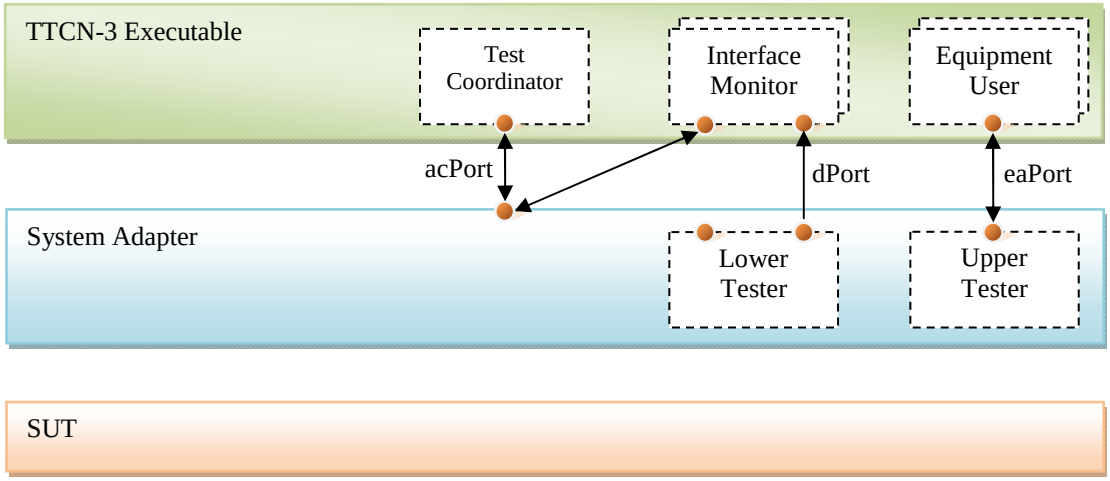


Figure 6: Abstract Port Associations

TTCN-3 components, i.e. the TTCN-3 TE, uses the Test System Interface (TSI) adapter configuration port `acPort` to perform general configuration as well as set filters in the lower tester component which is responsible for sending monitored messages to the TE. The port is also used to start and stop traffic capture. Note that every TTCN-3 component that would like to receive traffic has to request a filter setting from the adapter.

The TSI data port `dPort` is used by the SA to send to TTCN-3 component which have been captured during the monitoring of EUT communication and filtered.

The equipment access port `eaPort` is used by TTCN-3 components to request operation of equipment from at the System Adapter, e.g. to trigger for example the registration of a UE.

6.2.1 Adapter Configuration Primitives

Table 4 provides and over view all adapter configuration primitives expected to be supported by the ATS, their parameters and usage information. For more information the reader is referred to the TTCN-3 module `LibIot_TypesAndValues`.

Table 4: Adapter Configuration Primitives

Primitive	Parameters	Usage
GeneralConfigurationRequest GeneralConfigurationResponse	IP address and port of capture process, live vs. offline test execution, physical interfaces, recording vs. no recording, capture file offset, capture file or files to be merged	First message send to the adapter by ATS
SetFilterRequest SetFilterResponse	Protocol to be filtered IP address and port information related to interface	Can be sent by any component but not during capture
StartCaptureRequest StartCaptureResponse	None	Either sent after the general configuration or one or more filter requests
StopCaptureRequest StopCaptureResponse	None	Sent after start capture request

6.2.2 Upper Tester Primitives

Table 5 provides an overview of all equipment primitives expected to be supported by the ATS, their parameters and usage information. For more information the reader is referred to the TTCN-3 module `LibUpperTester`.

Table 5: Upper Tester Primitives

Primitive	Parameters	Usage
EquipmentOperationRequest EquipmentOperationResponse	command, parameter list	

6.2.3 TRI message encoding

All messages are exchanged via the TRI in encoded format including adapter configuration and equipment operation primitives. In order to be able to mix and match components from different vendors in a test system the following encoding rules have been defined for encoding adapter configuration and equipment operation messages:

- The message type is encoded in the first octet except for capturing messages which are pure raw data (see table 6 for details).
- Each information element of a message is encoded with < length><value> where < length> is always encoded on 2 octets.
- Text string values are kept as they are.
- Integer values are always encoded on 8 octets, using network byte order.
- Enumerated values are encoded in their integer representation using 1 octet.
- Lists of information elements are encoded using <number of parameters><{< length><value>}+ >, where <number of parameters> should use 2 octets and <length> <value> are encoded as described above.
- Sequences of information elements simply encoded as a concatenation of encoded information elements; note that the position of list information elements is assumed to be known, i.e. hardcoded.
- Union elements are encoded using <alternative index> in a single octet; the index starts at zero and the alternative definition order is assumed to be the same as in the TTCN-3 types defined in section X.
- Omitted information elements or values of length zero simply are encoded using <length> (or a <number of parameters> for the lists of information elements) set to '0000'H.

Table 6: Message type encoding

Message type	Octet Value Encoding
GeneralConfigurationReq	0x00
GeneralConfigurationRsp	0x01
SetFilterReq	0x02
SetFilterRsp	0x03
StartTrafficCaptureReq	0x04
StartTrafficCaptureRsp	0x05
StopTrafficCaptureReq	0x06
StopTrafficCaptureRsp	0x07
EquipmentOperationReq	0x08
EquipmentOperationRsp	0x09

6.2A Platform Adapter requirements

The ATS has no special requirements regarding timing. It assumes an implementation of timers using real time.

There are no external functions defined as part of this test suite.

6.3 Codec requirements

This test suite requires a TTCN-3 Coding/Decoding entity that supports encoding SIP and SDP TTCN-3 values into SIP text messages and SDP payloads, as well as vice versa. In addition, it requires similar support for DNS messages. This test suite also expects an adapter configuration and equipment operation request encoder as well as a response decoder. This CoDec should be implemented in conformance with the standard TTCN-3 Control Interface (TCI) [i.5].

6.3.1 Relevant RFCs

The CoDec part should support all RFCs supported by the TTCN-3 SIP and DNS library type structure:

- RFC 3261 [i.15] SIP: Session Initiation Protocol.
- RFC 3262 [i.16] Reliability of Provisional Responses in the Session Initiation Protocol (SIP).
- RFC 3265 [i.17] Session Initiation Protocol (SIP)-Specific Event Notification.
- RFC 3313 [i.18] Private Session Initiation Protocol (SIP) Extensions for Media Authorization.
- RFC 3323 [i.19] A Privacy Mechanism for the Session Initiation Protocol (SIP).
- RFC 3325 [i.20] Private Extensions to the Session Initiation Protocol (SIP) for Asserted Identity within Trusted Networks.
- RFC 3326 [i.21] The Reason Header Field for the Session Initiation Protocol (SIP).
- RFC 3327 [i.22] Session Initiation Protocol (SIP) Extension Header Field for Registering Non-Adjacent Contacts.
- RFC 3329 [i.23] Security Mechanism Agreement for the Session Initiation Protocol (SIP).
- RFC 3455 [i.24] Private Header (P-Header) Extensions to the Session Initiation Protocol (SIP) for the 3rd-Generation Partnership Project (3GPP).
- RFC 3515 [i.25] The Session Initiation Protocol (SIP) Refer Method.
- RFC 3608 [i.26] Session Initiation Protocol (SIP) Extension Header Field for Service Route Discovery During Registration.
- RFC 3841 [i.27] Caller Preferences for the Session Initiation Protocol (SIP).

- RFC 3891 [i.28] The Session Initiation Protocol (SIP) "Replaces" Header.
- RFC 3892 [i.29] The Session Initiation Protocol (SIP) Referred-By Mechanism.
- RFC 4028 [i.30] Session Timers in the Session Initiation Protocol (SIP).
- RFC 4244 [i.31] An Extension to the Session Initiation Protocol (SIP) for Request History Information.
- RFC 5009 [i.32] Private Header (P-Header) Extension to the Session Initiation Protocol (SIP) for Authorization of Early Media.

Some SIP message constructs reuse some headers defined in the HTTP protocol. Thus the following RFCs should be partially supported:

- RFC 2616 [i.33] Hypertext Transfer Protocol -- HTTP/1.1.
- RFC 2617 [i.34] HTTP Authentication: Basic and Digest Access Authentication.

Regarding the payload of the SIP messages, the CoDec should support encoding and decoding of message bodies in the SDP format.

- RFC 4566 [i.35] SDP: Session Description Protocol.

For DNS the following RFCs are relevant.

- RFC 1035 [i.36] DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION.
- RFC 2915 [i.37] The Naming Authority Pointer (NAPTR) DNS Resource Record.

6.3.2 SIP and SDP codec requirements

6.3.2.1 Omission of the delimiters

The TTCN-3 types in SIP library provide an abstract representation of the SIP messages which also maps the transfer syntax of the SIP messages. This mapping is meant to ease the semantic analysis of the messages and all the elements in the messages that do not carry any information on a semantic point of view (thus that are present only for syntactic purpose) are not represented. This includes:

- white space and new lines between the header fields;
- delimiters used for separating the fields (eg. : ; , ? & = @ or " in case of quoted strings).

On the decoding side, the CoDec should match these symbols to identify the fields delimited, then store the content of the fields into a TTCN-3 data structure and discard these delimiters. On the encoding side, the CoDec should add all the necessary whitespaces and punctuations between the fields so as to produce a syntactically correct SIP message.

EXAMPLE: The code below shows an example of the representation of a Via message header in the raw format and its corresponding value using the TTCN-3 types in the SIP library.

Raw format:

```
Via: SIP/2.0/UDP
    192.0.2.1;branch=z9hG4Bk
```

Corresponding TTCN-3 value:

```
{
  fieldName := VIA_E
  viaBody := {{
    sentProtocol := {
      protocolName := "SIP",
      protocolVersion := "2.0",
      transport := "UDP",
      sentBy := {
        host := "192.0.2.1",
        portField := {
          viaParams := {{
            id := "branch",
            paramValue := "z9hG4bK"
}}}}}}}}
```

6.3.2.2 Normalisation

Some constructs in the SIP message format allow several representations of the same message that are semantically equivalent but that have a different syntax:

- The most common headers names can be replaced with a one-letter alias to shorten the messages. For example, the header name "From:" can simply be represented as "f:".
- The characters in a quoted string enclosed with double quotes (") may be escaped by a preceding backslash character (\).
- Most field values that are not enclosed within a quoted string may contain escaping sequences starting with a percent character (%) and followed by two hexadecimal digits coding the binary value of the character.

In order to ease the analysis of the messages received in the abstract test suite, these constructs should be normalised. Thus two messages that are syntactically different but semantically equivalent will produce exactly the same TTCN-3 value. Following this approach:

- The two possible variants for a header name will map to the same enumeration value (e.g. the "From:" and "f:" header names will both map to the "FROM_E" header value).
- All the escaping characters (\) in quoted strings will be removed. Note that this does not raise any operational issue since the enclosing quotes (") were removed and are no longer needed for delimiting the string.
- All the escape sequences (%xx) outside quoted strings will be replaced with the corresponding character if the character is a displayable character (in the 7-bit ASCII set).

Additionally the SIP message format is encoded using the Unicode UTF-8 character set. This encoding is identical to standard ASCII encoding for the ASCII character set but different for any characters which go beyond the ASCII range. Since SIP type definitions in LibSip map to the TTCN-3 charstring type and not the universal charstring, the type system cannot handle advanced UTF-8 encoded strings.

6.3.2.3 Other requirements

According to the conventions used for structuring the messages in the SIP message, the following considerations should be taken into account:

- Optional fields that may contain multiple values are represented in TTCN-3 with an "optional" field containing a "record of" or a "set of" structured type. In the case no options are present, the field should be omitted (instead of being present and containing an empty list).
- The SIP messages contain an additional field named "payload" (which is distinct from the "messageBody" field). This field is meant to contain the whole raw SIP message represented in the value. Its purpose is only for debugging purpose to provide a textual representation of the message in the TTCN-3 environment. Its content is always ignored in the abstract test suite. The CoDec should handle this field as follows:
  - when encoding a message, the payload field should be ignored by the CoDec;
  - when decoding a message, the CoDec should fill the payload field with the whole SIP message in the textual format (as received from the System Adapter). Note this field is a 7-bit charstring, therefore the non-displayable characters should be replaced or escaped to avoid that the TTCN-3 environment report any error.
- The message headers in the SIP messages are syntactically represented as a list of headers. However since the position of headers is not significant (apart from headers that may appear multiple times) and since most of the headers can occur only once in a SIP messages, it was decided to represent the message headers as a single TTCN-3 set containing one field per header type. The filed type of the headers that can appear multiple times contain a "record of" for storing the successive occurrences of the header. This structure does not reflect accurately the message structure, however it easy considerably the semantic analysis of the message (a given header can be accessed directly as a field in the set instead of having to find it inside a list of headers). The CoDec is required to accommodate this representation.

6.4 ATS limitations

6.4.1 Authentication and Security

In case of full IMS, security test monitors will not be able to decode messages. In this case the IPsec authentication/security should be disabled if possible or null authentication has to be used. Otherwise there is no way to decode the messages. Otherwise Interface Monitor test components should be disabled from test execution via respective module parameter settings.

6.4.2 Automation of operation user equipment

The TTCN-3 code of this ATS has been implemented to support automatic operation of user equipment. Upper tester adaptation is currently limited output of equipment operation commands to a text terminal for human user equipment operators. Adapters for a direct integration with specific IMS soft clients is possible but not yet implemented.

7 Test execution aspects

7.1 Live versus offline test execution

Automated interoperability testing can be used with either live (or "in real time") or an offline test execution settings. In the live case the test suite operates user equipment (or instructs to operate it) and analysis a live capture. In the offline case it is simply assumed that equipment operation has been performed manually and that relevant traffic on all interfaces has been captured in one or more traffic capture trace files, e.g. in a PCAP file in the case of IMS NNI IOT.

Interesting or valuable results arise when test cases fail, but often, e.g. due to the challenging testing conditions at an interoperability event which only offers a very limited amount of time or incorrect EUT interface information, it is not possible to analyze results and find out the reasons for a failure especially of conformance assessments in real time. For this reason it is a requirement to an automated interoperability test system to provide a test execution mode that allows to work based purely on interface traces.

This test suite supports both approaches. The default execution mode is e_offline. The test execution module parameter PX_TEST_EXECUTION has to be set to the value to "e_live" to enable the use of the real time mode.

7.2 Unavailable monitored interfaces

During or after an interoperability test, one or more EUT interfaces may not be available for monitoring during test execution analysis. This test suite uses module parameters to indicate the availability of each monitored interface in a test and assumes that by default all interfaces are available.

In order to deactivate either an interface monitor component its respective PIXIT, e.g. PIXIT_ISC_A_AVAILABLE should be set to "false". Note that the PIXIT settings are taken into account in all test executions. The effect of removing an interface is that it is not included in the verdict resolution. In the handling of the conformance verdict it results however to a reduction of the pass into an inconclusive verdict in case the test contains one or more explicit checks on a disabled monitored interface.

7.3 Test case selection

Different criteria have to be considered when selecting them for execution. Tests can be grouped for execution according to:

- the functionality they assess, e.g. IMS registration, call, or messaging;
- their test configuration which will minimize the effort to reconfigure the test system between tests;
- pre-test conditions are which allows to skip an entire group of tests in case a pre-test condition cannot be fulfilled;
- an assigned test priority.

Note that the above list is not exhaustive. Based on the aim of testing and available resource, alternative ways of grouping can be added. Also, the test selection can be based on more than one grouping.

Annex A:
Electronic annex

The IMS IOT ATS has been produced using the Testing and Test Control Notation (TTCN) according to ES 201 873-1 [i.6].

It is contained in archive tr_102788v010101p0.zip which accompanies the present document.

History

Document history		
V1.1.1	January 2010	Publication