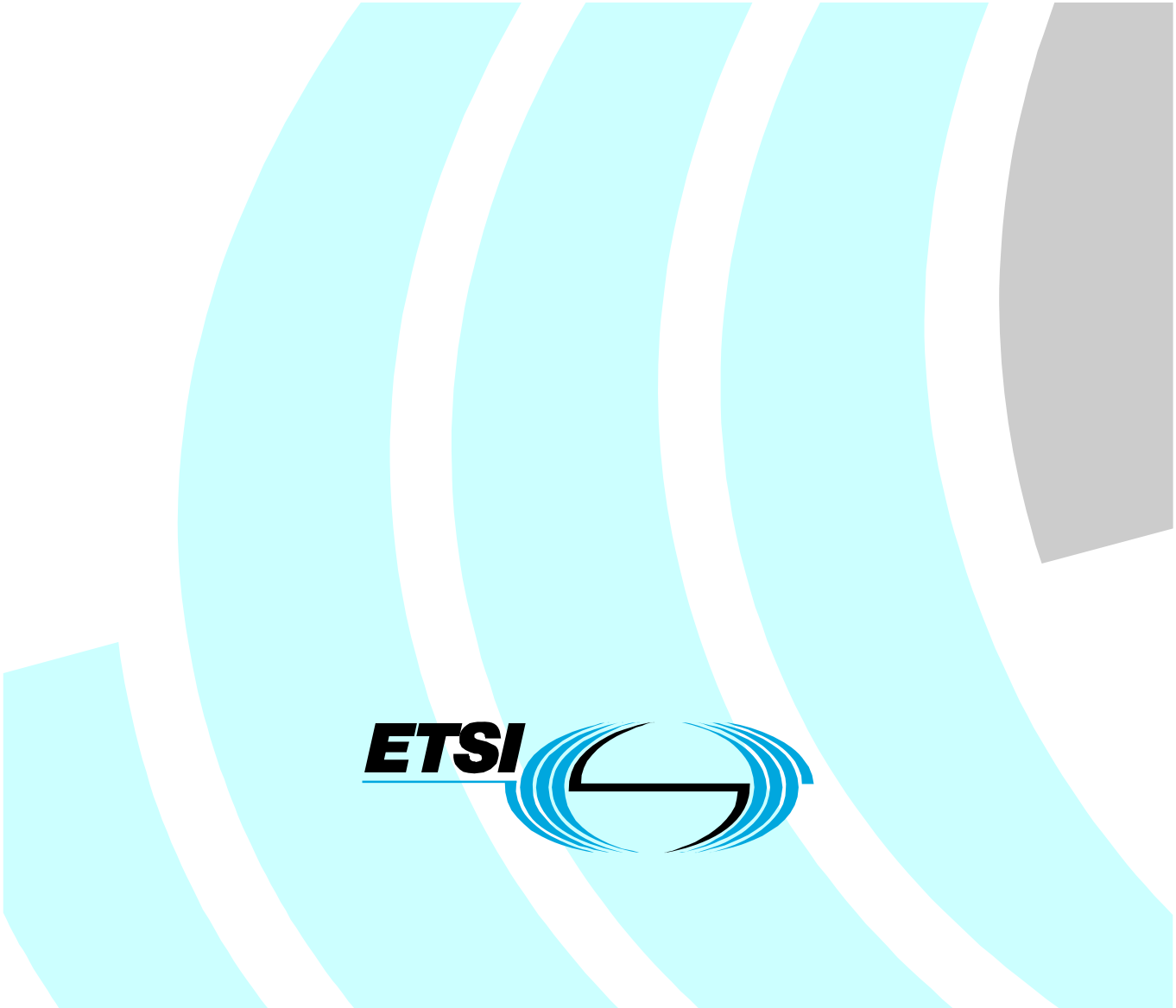
ETSI TS 186 011-3 V3.1.1 (2011-06)

Technical Specification

IMS Network Testing (INT); IMS NNI Interoperability Test Specifications; Part 3: Abstract Test Suite (ATS) and partial Protocol Implementation eXtra Information for Testing (PIXIT)



Reference
RTS/INT-00032-3

Keywords
IMS, interworking, NNI, ATS, 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

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 Specification (TS) has been produced by ETSI Technical Committee IMS Network Testing (INT).

The present document is part 3 of a multi-part deliverable covering the IMS NNI Interoperability Test Specifications, as identified below:

Part 1: "Test Purposes for IMS NNI Interoperability";

Part 2: "Test descriptions for IMS NNI Interoperability";

Part 3: "Abstract Test Suite (ATS) and partial Protocol Implementation eXtra Information for Testing (PIXIT)".

1 Scope

The present document describes the Abstract Test Suite (ATS) to test interoperability at IMS NNI for IP multimedia call control protocol based on TS 124 229 [1]. The ATS has been specified on the basis of the Test Descriptions for IMS NNI interoperability testing presented in TS 186 011-2 [3]. It defines a TTCN-3 framework as well as codec and adapter requirements for analysing interoperability test execution traces generated from the manual or automatic execution of IMS interoperability tests.

The scope of this ATS is not to cover all requirements specified in TS 124 229 [1]. It only assesses requirements that are observable at the NNI between two IMS core network implementations specified in TS 186 011-1 [2].

2 References

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the reference document (including any amendments) applies.

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 necessary for the application of the present document.

[1] ETSI TS 124 229 (V8.10.0): "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 8.10.0 Release 8)".

[2] ETSI TS 186 011-1 (V3.1.1): "IMS Network Testing (INT); IMS NNI Interoperability Test Specifications; Part 1: Test Purposes for IMS NNI Interoperability".

[3] ETSI TS 186 011-2 (V3.1.1): "IMS Network Testing (INT); IMS NNI Interoperability Test Specifications; Part 2: Test Description for IMS NNI Interoperability".

[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)".

[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)".

[6] 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".

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

2.2 Informative references

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] ETSI EG 202 568: "Methods for Testing and Specification (MTS); Internet Protocol Testing (IPT); Testing: Methodology and Framework".

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

[i.3] IETF RFC 3262: "Reliability of Provisional Responses in the Session Initiation Protocol (SIP)".

[i.4] IETF RFC 3265: "Session Initiation Protocol (SIP)-Specific Event Notification".

[i.5] IETF RFC 3313: "Private Session Initiation Protocol (SIP) Extensions for Media Authorization".

[i.6] IETF RFC 3323: "A Privacy Mechanism for the Session Initiation Protocol (SIP)".

[i.7] IETF RFC 3325: "Private Extensions to the Session Initiation Protocol (SIP) for Asserted Identity within Trusted Networks".

[i.8] IETF RFC 3326: "The Reason Header Field for the Session Initiation Protocol (SIP)".

[i.9] IETF RFC 3327: "Session Initiation Protocol (SIP) Extension Header Field for Registering Non-Adjacent Contacts".

[i.10] IETF RFC 3329: "Security Mechanism Agreement for the Session Initiation Protocol (SIP)".

[i.11] IETF RFC 3455: "Private Header (P-Header) Extensions to the Session Initiation Protocol (SIP) for the 3rd-Generation Partnership Project (3GPP)".

[i.12] IETF RFC 3515: "The Session Initiation Protocol (SIP) Refer Method".

[i.13] IETF RFC 3608: "Session Initiation Protocol (SIP) Extension Header Field for Service Route Discovery During Registration".

[i.14] IETF RFC 3841: "Caller Preferences for the Session Initiation Protocol (SIP)".

[i.15] IETF RFC 3891: "The Session Initiation Protocol (SIP) "Replaces" Header".

[i.16] IETF RFC 3892: "The Session Initiation Protocol (SIP) Referred-By Mechanism".

[i.17] IETF RFC 4028: "Session Timers in the Session Initiation Protocol (SIP)".

[i.18] IETF RFC 4244: "An Extension to the Session Initiation Protocol (SIP) for Request History Information".

[i.19] IETF RFC 5009: "Private Header (P-Header) Extension to the Session Initiation Protocol (SIP) for Authorization of Early Media".

[i.20] IETF RFC 2616: "Hypertext Transfer Protocol -- HTTP/1.1".

[i.21] IETF RFC 4566: "SDP: Session Description Protocol".

[i.22] IETF RFC 1035: "Domain names - implementation and specification".

[i.23] IETF RFC 2915: "The Naming Authority Pointer (NAPTR) DNS Resource Record".

[i.24] IETF RFC 2617: "HTTP Authentication: Basic and Digest Access Authentication".

[i.25] ETSI EG 202 810: "Methods for Testing and Specification (MTS); Automated Interoperability Testing; Methodology and Framework".

3 Abbreviations

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

AS	(IMS) Application Server
ATS	Abstract Test Suite
CSCF	Call Session Control Function
IBCF	Interconnection Border Control Gateway
I-CSCF	Interrogating CSCF
IMS	IP Multimedia Subsystem
IP	Internet Protocol
MTC	Main Test Component

NNI	Network to Network Interface
PCO	Point of Control and Observation
P-CSCF	Proxy CSCF
PO	Point of Observation
PTC	Parallel Test Component
S-CSCF	Serving CSCF
SIP	Session Initiation Protocol
SUT	System Under Test
TD	Test Description
TP	Test Purpose
TSI	Test System Interface
TTCN-3	Testing and Test Control Notation 3
UE	User Equipment

4 Overview

4.1 Network architecture

The ATS is defined to observe the SIP communication at the Gm, Mx, Ici and ISC interface of two IMS core networks for interoperability testing. Figure 1 shows a general architecture of two IMS core networks including the related interfaces.

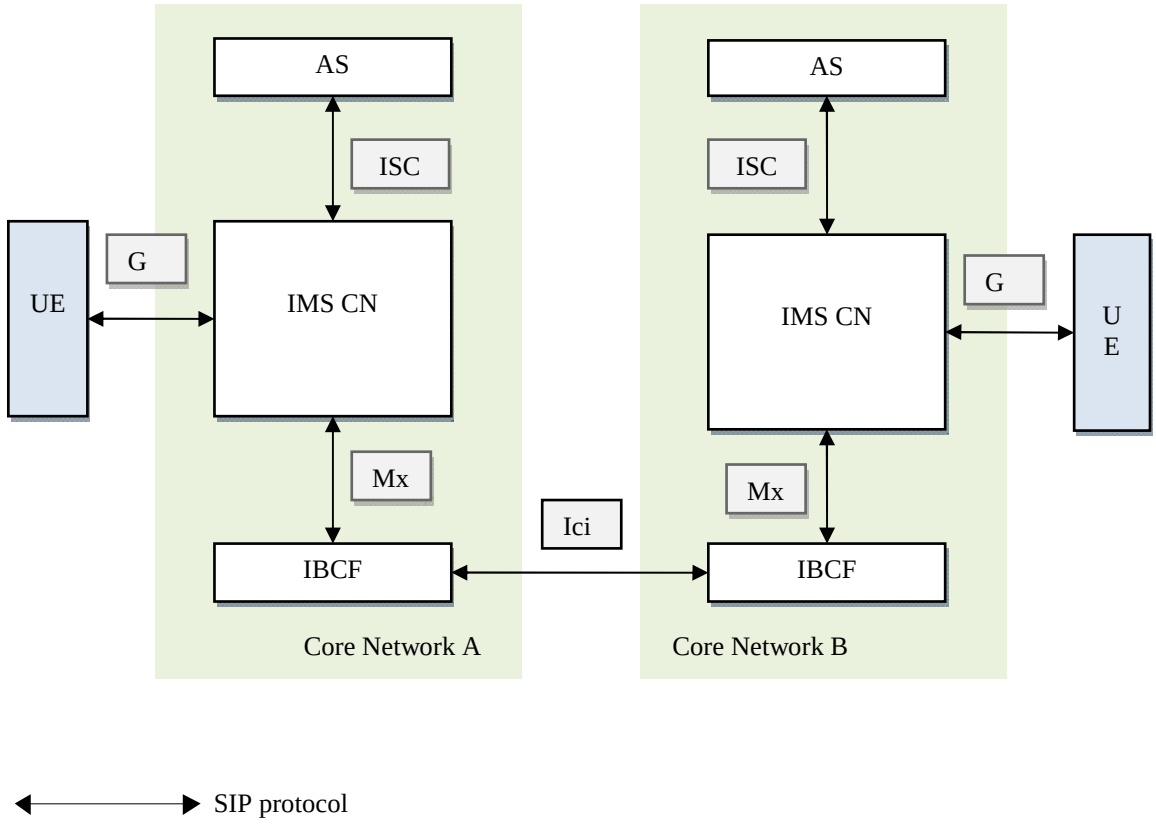


Figure 1: Network Architecture

4.1.1 Core IMS Nodes

The P-CSCF, S-CSCF, I-CSCF and the IBCF are considered to be within a "black box" for testing purposes, i.e. the System Under Test (SUT). Interfaces within the IMS are considered internal and not observable for testing purposes except the Mx interface. The Mx and the Ici interface between two IMS core networks is used as point of observation (PO) for NNI interoperability tests.

4.1.2 External IMS Nodes

4.1.2.1 UE

The UE is considered to act as stimulus node in this test specification. The Gm interface between the P-CSCF and the UE is used as a Point of Control and Observation (PCO) for NNI interoperability tests.

4.1.2.2 AS

The Application Server (AS) is considered to act as a stimulus node in this test specification. The ISC interface between the S-CSCF and the AS is used as Point of Control and Observation (PCO) for NNI interoperability tests.

5 Test configuration

The test configuration is described in detail in TS 186 011-2 [3].

Test configurations have been defined in [3] 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 2. For detailed discussion of the abstract test system interface, the reader is referred to clause 6.2.

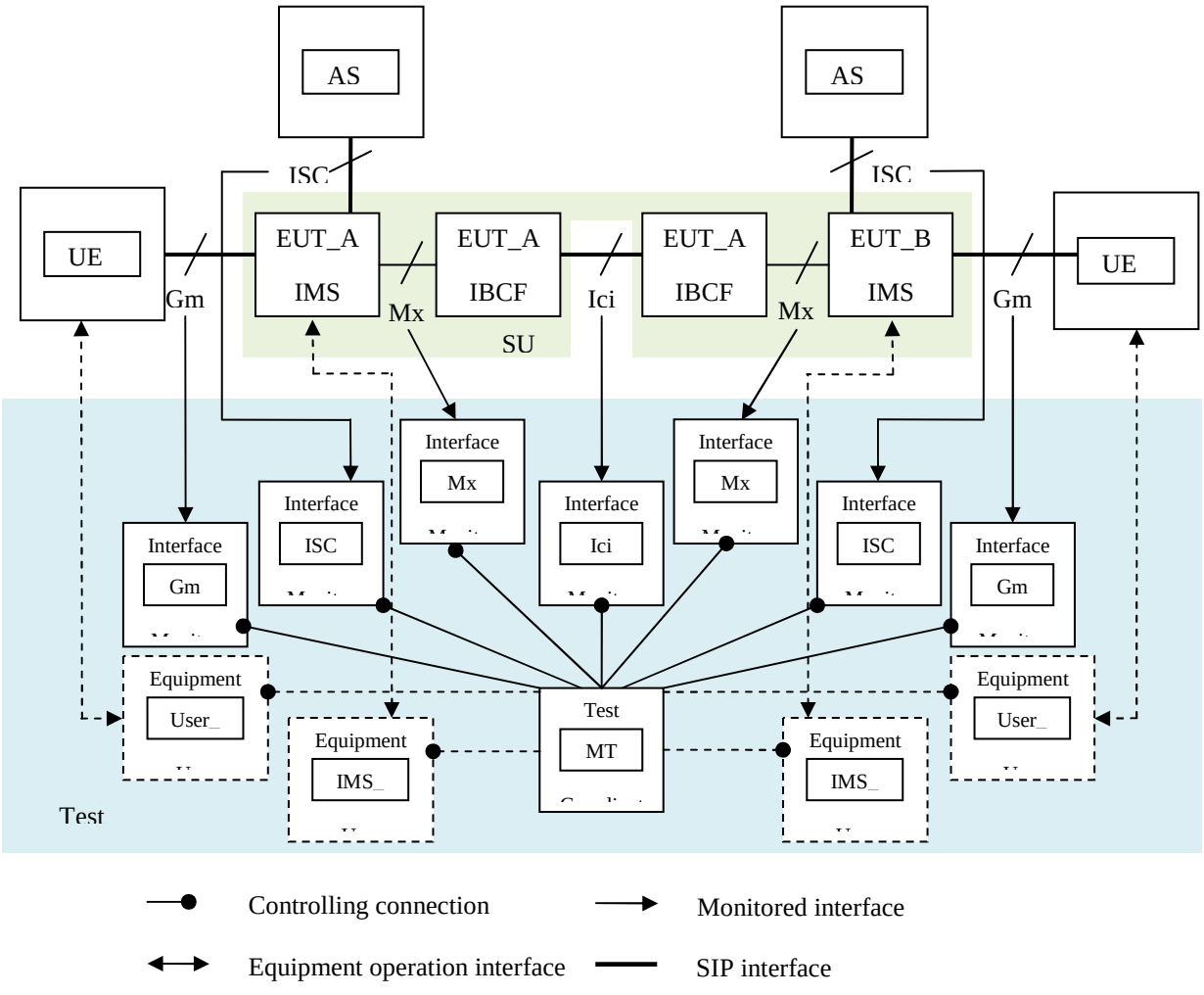


Figure 2: Example IMS NNI interoperability test system configuration

The test system configuration is based on the general TTCN-3 test system architecture specified in ES 201 873-5 [4] and ES 201 873-6 [5] as well as on the concepts stated in EG 202 810 [i.25]. Note that the figure 2 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 2 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.

6 Test design

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

The ATS is specified using TTCN-3 [7]. 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.

6.1 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.1] 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 4) 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 1 specifies the naming guidelines for each construct of the TTCN-3 language indicating the recommended prefix and capitalization.

Table 1: 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	Note 1
Message template with wildcard or matching expression	Lower-case initial letter	mw_	mw_response	Note 2
Modifying message template	Lower-case initial letter	md_	md_response	Note 1
Modifying message template with wildcard or matching expression	Lower-case initial letter	mdw_	mdw_reponse	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	Note 3
Variable (defined within component type)	Lower-case initial letter	vc_	vc_systemName	
Timer (defined locally)	Lower-case initial letter	t_	t_wait	

Language element	Naming convention	Prefix	Example	Notes
Timer (defined within component type)	Lower-case initial letter	tc_	tc_authMin	
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 7.5.

6.2 TTCN-3 language version

This test suite has been developed based on the concepts available in version 4.4.1 of the TTCN-3 core language defined in ES 201 873-1 [7]. 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.

6.3 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 7.9. 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.

6.4 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.

6.5 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.

6.6 Handling of proprietary interfaces

Equipment user test components (see figure 2) 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 2) 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: For 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.

6.7 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 shall 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.

6.8 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 [6]. 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 2.

Table 2: 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 this 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.

6.9 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"

7 Test system

7.1 Test system architecture

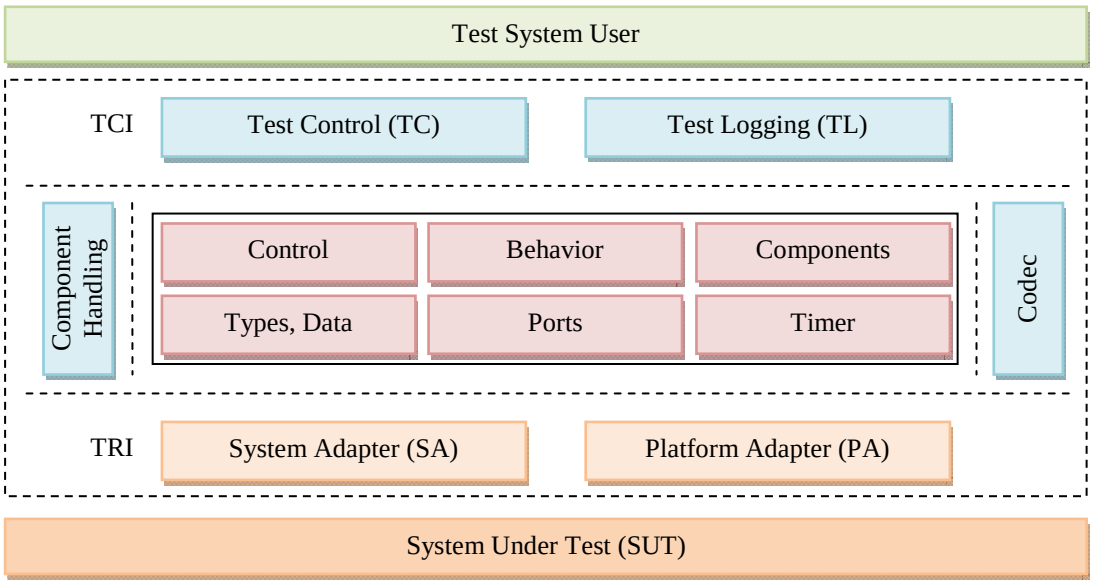


Figure 3: Abstract Test System Architecture

Figure 3 shows the abstract test system architecture. It shows a test system (TS) that is compliant to [4] and [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 [4] and [5].

7.1.1 SUT adapter requirements

Figure 4 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.

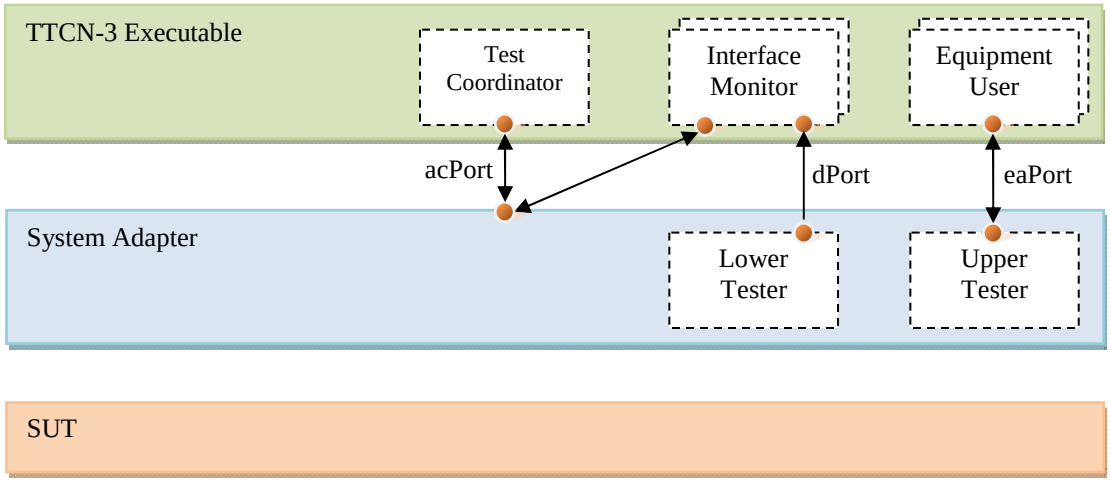


Figure 4: Abstract Port Associations

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.

7.1.2 Adapter Configuration Primitives

Table 3 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 3: 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

7.1.3 Upper Tester Primitives

Table 4 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 4: Upper Tester Primitives

Primitive	Parameters	Usage
EquipmentOperationRequest EquipmentOperationResponse	command, parameter list	

7.1.4 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 below table 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> shall 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 5: 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

7.2 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.

7.3 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 shall be implemented in conformance with the standard TTCN-3 Control Interface (TCI) [5].

7.3.1 Relevant RFCs

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

- RFC 3261: "SIP: Session Initiation Protocol".
- RFC 3262: "Reliability of Provisional Responses in the Session Initiation Protocol (SIP)".
- RFC 3265: "Session Initiation Protocol (SIP)-Specific Event Notification".
- RFC 3313: "Private Session Initiation Protocol (SIP) Extensions for Media Authorization".
- RFC 3323: "A Privacy Mechanism for the Session Initiation Protocol (SIP)".
- RFC 3325: "Private Extensions to the Session Initiation Protocol (SIP) for Asserted Identity within Trusted Networks".
- RFC 3326: "The Reason Header Field for the Session Initiation Protocol (SIP)".
- RFC 3327: "Session Initiation Protocol (SIP) Extension Header Field for Registering Non-Adjacent Contacts".

- RFC 3329: "Security Mechanism Agreement for the Session Initiation Protocol (SIP)".
- RFC 3455: "Private Header (P-Header) Extensions to the Session Initiation Protocol (SIP) for the 3rd-Generation Partnership Project (3GPP)".
- RFC 3515: "The Session Initiation Protocol (SIP) Refer Method".
- RFC 3608: "Session Initiation Protocol (SIP) Extension Header Field for Service Route Discovery During Registration".
- RFC 3841: "Caller Preferences for the Session Initiation Protocol (SIP)".
- RFC 3891: "The Session Initiation Protocol (SIP) "Replaces" Header".
- RFC 3892: "The Session Initiation Protocol (SIP) Referred-By Mechanism".
- RFC 4028: "Session Timers in the Session Initiation Protocol (SIP)".
- RFC 4244: "An Extension to the Session Initiation Protocol (SIP) for Request History Information".
- RFC 5009: "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: "Hypertext Transfer Protocol -- HTTP/1.1".
- RFC 2617: "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: "SDP: Session Description Protocol".

For DNS the following RFCs are relevant.

- RFC 1035: "Domain names - implementation and specification".
- RFC 2915: "The Naming Authority Pointer (NAPTR) DNS Resource Record".

7.3.2 SIP and SDP codec requirements

7.3.2.1 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 shall 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 shall 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"
}}}}}}}
```

7.3.2.2 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.

7.3.2.3 Other requirements

According to the conventions used for structuring the messages in the SIP message, the following considerations shall 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 shall 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 shall handle this field as follows:
  - when encoding a message, the payload field shall be ignored by the CoDec
  - when decoding a message, the CoDec shall 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 shall 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.

8 Test execution

8.1 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.

8.2 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.

8.3 Test case selection

When selecting test cases, different aspects have to be considered. Test may be selected based on a grouping. Tests can be grouped as follows:

- functionality to test: If a certain functionality need to be tested, the test belonging to this functionality should be selected, e.g. in IMS registration, call, or messaging;

- test configurations: If the test are selected based on test configurations, the effort to reconfigure the test system is as minimal as possible;
- pre-test conditions: This is suggested if certain pre-test-conditions are repeatable in the test description, but , cannot be fulfilled. Therefore, these tests would have to be skipped;
- priority: If a priority has been assign to the tests, the tests should be selected based on the priority.

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

8.4 PIXIT

8.4.1 LibIot_PIXITS

PX_TTCN3_VERDICT	PIXIT defines which verdict (E2E or conformance)is to be kept track of with the TTCN-3 verdict.
PX_MAX_MSG_WAIT	Maximum time limit used by monitor component for waiting for expected incoming messages.
PX_PRODUCTS	Example of module parameter based entry of EUT interface information for all products participating in an interoperability event.
PX_EUT_A	Selects product based on index in PX_PRODCUTS vendor list for EUT_A.
PX_EUT_B	Selects product based on index in PX_PRODCUTS vendor list for EUT_B.
PX_EUT_C	Selects product based on index in PX_PRODCUTS vendor list for EUT_C.
PX_EUT_B_B2	Selects product based on index in PX_PRODCUTS vendor list for EUT_B_B2.
PX_AVAILABLE_INTERFACES	Selects if interfaces should be considered in the evaluation interfaceName needs to be consistent to AtsImsIot_TestConfiguration.
PX_EUT_TRIGGER_RESPONSE	Maximum time limit used by trigger component for waiting for EUT response after command has been sent.
PX_IOT_PCAP_SESSIONS_PATH	In case of offline mode, it defines the path where all sessions's Pcap files are located.
PX_IOT_RECORD_MODE	Defines if the record traffic capture mode must be activated or not.
PX_IOT_FILE_MERGE_LIST	Defines list of the files to merge.
PX_IOT_FILE_MERGE_PATH	Defines the location of the files to merge.
PX_IOT_MERGE_TOOL_PATH	Defines the location of the files to merge.
PX_IOT_TIMESTAMP_OFFSET	Defines the time stamp offset to start playing record traffic capture file.
PX_IOT_IFACES	List of the network interfaces to monitor.
PX_IOT_EUTs_IFACE_INFO_LIST	List of EUT network interfaces.

8.4.2 LibIms_UpperTester

PX_IMS_USER_DATA	Example of module parameter based entry of EUT interface information for all products particpating in an interoperability event.

Annex A (normative): Zip file with TTCN-3 code

A.1 The ATS in TTCN-3 core (text) format

This ATS has been produced using the Testing and Test Control Notation (TTCN) according to ES 201 873-1 [7].

The TTCN-3 core (text) representation corresponding to this ATS is contained in an ASCII file(s) (IMS_TestSystem.**ttcn3** contained in archive ts_18601103v030101p0.zip) which accompanies the present document.

Where an ETSI Abstract Test Suite (in TTCN-3) is published in both core and tabular format these two forms shall be considered equivalent. In the event that there appears to be syntactical or semantic differences between the two then the problem shall be resolved and the erroneous format (whichever it is) shall be corrected.

History

Document history		
V2.2.1	September 2009	Publication
V3.1.1	June 2011	Publication