ETSI TS 103 268-1 V1.1.1 (2017-04)

SmartM2M;
Smart Appliances Ontology and Communication Framework Testing;
Part 1: Testing methodology

Reference
DTS/SmartM2M-103 268-1 SAP_tst

Keywords
data, data sharing, IoT, M2M, Model,
multi service testing, oneM2M, ontology, 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

The present document can be downloaded from:
http://www.etsi.org/standards-search

The present document may be made available in electronic versions and/or in print. The content of any electronic and/or print versions of the present document shall not be modified without the prior written authorization of ETSI. In case of any existing or perceived difference in contents between such versions and/or in print, the only prevailing document is the print of the Portable Document Format (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
https://portal.etsi.org/TB/ETSIDeliverableStatus.aspx

If you find errors in the present document, please send your comment to one of the following services:
https://portal.etsi.org/People/CommiteeSupportStaff.aspx

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 (https://ipr.etsi.org/).

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 Smart Machine-to-Machine communications (SmartM2M).

The present document is part 1 of a multi-part deliverable covering Conformance test specifications for Smart Appliances Ontology and Communication Framework Testing as identified below:

Part 1: "Testing methodology";

Part 2: "Protocol Implementation Conformance Statement (PICS) pro forma";

Part 3: "Test Suite Structure and Test Purposes (TSS & TP)";

Part 4: "Abstract Test Suite (ATS) and Protocol Implementation eXtra Information for Testing (PIXIT)".

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

1 Scope

The scope of the present document is to support Smart Appliance common ontology and communication framework testing needs. It specifies a global methodology for testing for Smart Appliances, based oneM2M specifications. It analyses the overall testing needs and identifies and defines the additional documentation required.

The testing framework proposed in the present document provides methodology for development of conformance and interoperability test strategies, test systems and the resulting test specifications for SAP.

2 References

2.1 Normative 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 referenced document (including any amendments) applies.

Referenced documents which are not found to be publicly available in the expected location might be found at https://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.

The following referenced documents are necessary for the application of the present document.

[1] ETSI TS 103 264: "SmartM2M; Smart Appliances; Reference Ontology and oneM2M Mapping".

[2] ETSI TS 103 267: "SmartM2M; Smart Appliances; Communication Framework".

[3] ETSI TS 118 101: "oneM2M; Functional Architecture (oneM2M TS-0001)".

[4] ETSI TS 118 104: "oneM2M; Service Layer Core Protocol Specification (oneM2M TS-0004)".

[5] ETSI TS 118 112: "oneM2M; Base Ontology (oneM2M TS-0012)".

[6] ETSI TS 118 108: "oneM2M; CoAP Protocol Binding (oneM2M TS-0008)".

[7] ETSI TS 118 109: "oneM2M; HTTP Protocol Binding (oneM2M TS-0009)".

[8] ETSI TS 118 110: "oneM2M; MQTT Protocol Binding (oneM2M TS-0010)".

[9] ETSI TS 118 120: "oneM2M; WebSocket Protocol Binding (oneM2M TS-0020)".

2.2 Informative 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 referenced document (including any amendments) applies.

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

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 202 798: "Intelligent Transport Systems (ITS); Testing; Framework for conformance and interoperability testing".

[i.2] ISO/IEC 9646 (all parts): "Information technology - Open Systems Interconnection - Conformance testing methodology and framework".

[i.3] ETSI ES 201 873 (all parts): "Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3".

[i.4] ETSI EG 202 237: "Methods for Testing and Specification (MTS); Internet Protocol Testing (IPT); Generic approach to interoperability testing".

[i.5] ETSI EG 201 058: "Methods for Testing and Specification (MTS); Implementation Conformance Statement (ICS) pro forma style guide".

[i.6] ETSI ETR 266: "Methods for Testing and Specification (MTS); Test Purpose style guide".

[i.7] ETSI ETS 300 406: "Methods for Testing and Specification (MTS); Protocol and profile conformance testing specifications; Standardization methodology".

[i.8] ISO 10746 (all parts): "Information technology - Open Distributed Processing - Reference model".

[i.9] ETSI TS 103 268-2:"SmartM2M; Smart Appliances Ontology and Communication Framework Testing; Part 2: Protocol Implementation Conformance Statement (PICS) proforma".

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in ISO 10746 [i.8], ETSI TS 103 264 [1], ETSI TS 103 267 [2], ETSI TS 118 101 [3], ETSI TS 118 104 [4] and the following apply:

conformance testing: process for testing that an implementation is compliant with a protocol standard, which is realized by test systems simulating the protocol with test scripts executed against the implementation under test

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

testing framework: document providing guidance and examples necessary for the development and implementation of a test specification

conformance: compliance with requirements specified in applicable standards ISO/IEC 9646 [i.2]

Device Under Test (DUT): combination of software and/or hardware items which implement the functionality of standards and interact with other DUTs via one or more reference points

Implementation Under Test (IUT): implementation of one or more Open Systems Interconnection (OSI) protocols in an adjacent user/provider relationship, being the part of a real open system which is to be studied by testing (ISO/IEC 9646-1 [i.2])

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

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

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

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

NOTE 1: Once a DUT has been successfully tested against a QE, it may be considered to be a QE, itself.

NOTE 2: Once a QE is modified, it loses its status as QE and becomes again a DUT.

test case: specification of the actions required to achieve a specific test purpose, starting in a stable testing state, ending in a stable testing state and defined in either natural language for manual operation or in a machine-readable language (such as TTCN-3) for automatic execution

test purpose: description of a well-defined objective of testing, focussing on a single interoperability requirement or a set of related interoperability requirements

3.2 Abbreviations

For the purposes of the present document, the abbreviations given in ISO 10746 [i.8], ETSI TS 103 268-2 [i.9], ETSI TS 118 101 [3], ETSI TS 118 104 [4] and the following apply:

API	Application Programming Interface
ATS	Abstract Test Suite
DUT	Device Under Test
EUT	Equipment Under Test
IFS	Interoperable Features Statement
IUT	Implementation Under Test
IWF	InterWorking Function
PICS	Protocol Implementation Conformance Statement
PIXIT	Protocol Implementation eXtra Information for Testing
QE	Qualified Equipment
RP	Reference Point
SAP	Smart Appliance
SUT	System Under Test
TP	Test Purpose
TSS	Test Suite Structure
TST	Testing

4 Introduction to the SAP testing methodology

The present document provides:

- Identification of the implementations under test (IUT) for conformance testing and the device under test (DUTs) for interoperability, i.e. answering the question "what is to be tested".
- Definition of the applicable test procedures, i.e. answering the question "how is to be tested".
- Definition of the procedure for development of test specifications and deliverables (for instance: TSS & TP, TP pro forma, TTCN-3 test suite and documentation).

Figure 4-1 illustrates the SAP M2M testing framework and the interactions with M2M base standards and SAP M2M test specifications. The SAP M2M testing framework is based on concepts defined in ISO/IEC 9646 [i.2], TTCN-3 [i.3], ETSI EG 202 237 [i.4] and ETSI EG 202 798 [i.1].

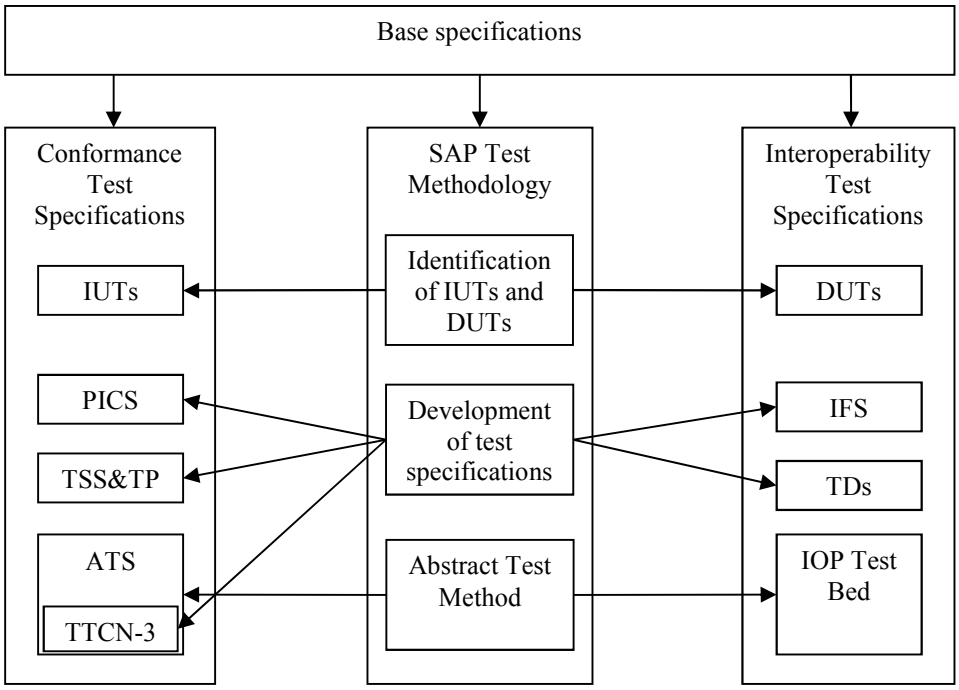

Figure 4-1: SAP M2M testing methodology interactions

ETSI test specifications are usually developed for a single base protocol standard or for a coherent set of standards. As such, it is possible to follow the methodology specified for conformance test development in ISO/IEC 9646-1 [i.2] without much difficulty. However, M2M and Smart Appliance testing requirements are, in many cases, distributed across a wide range of documents and, thus, an adaptation of the ISO/IEC 9646 [i.2] approach to test development is necessary. Also, for readability, consistency and to ease reusability of TTCN-3 code it is necessary to apply some guidelines on the use of TTCN-3.

It is this approach that is referred to as the "5Smart Appliance testing framework".

As its name implies, the framework is oriented towards the production of Test specifications. The Smart Appliance testing Framework comprises:

- A documentation structure:
  - Catalogue of requirements (PICS or IFS).
  - Test Suite Structure (TSS).
  - Test Purposes:
    - Conformance.
    - Interoperability.
- A methodology linking the individual elements of a test specification together:
  - Style guidelines and examples.
  - Naming conventions.
  - A structured notation for TP.
  - Guidelines on the development of TTCN-3 Test Cases (TCs).
  - Guidelines on the use of tabulated English Test Descriptions (TDs).

5 Conformance testing

5.1 Introduction

The following clauses show how to apply the ETSI conformance testing methodology to SAP M2M in order to properly produce SAP M2M conformance test specifications.

The Conformance testing can show that a product correctly implements a particular standardized protocol, that is, it establishes whether or not the implementation under test meets the requirements specified for the protocol itself.

For example, it will test protocol message contents and format as well as the permitted sequences of messages. In that context, tests are performed at open standardized interfaces that are not (usually) accessible to an end user, and executed by a dedicated test system that has full control of the system under test and the ability to observe all incoming and out coming communications; the high degree of control of the test system over the sequence and contents of the protocol messages allows to test both valid and invalid behaviour.

Figure 5.1-1: Conformance testing

Conformance test specifications should be produced following the methodology described in ISO/IEC 9646-1 [i.2]. In summary, this methodology begins with the collation and categorization of the requirements to be tested into a tabular form which is normally referred to as the "Protocol Implementation Conformance Statement" (PICS). Each PICS relates to a specific protocol standard. In those cases where the requirements are distributed across a large number of documents there may be very little benefit in producing an individual PICS for each document. Consequently, the requirements should be collected together and categorized in a single document, referred as the Requirements Catalogue. The present document could be structured as an overall PICS covering the requirements of all the relevant specifications.

For each requirement in the catalogue, one or more tests should be identified and classified into a number of groups which will provide a structure to the overall test suite (TSS). A brief Test Purpose (TP) should then be written for each identified test and this should make it clear what is to be tested but not how this should be done. Although not described or mandated in ISO/IEC 9646-1 [i.2], in many situations (particularly where the TPs are complex) it may be desirable to develop a Test Description (TD) for each TP. The TD describes in plain language (often tabulated) the actions required to reach a verdict on whether an implementation passes or fails the test. Finally, a detailed Test Case (TC) is written for each TP. In the interests of test automation, TCs are usually combined into an Abstract Test Suite (ATS) using a specific testing language such as TTCN-3.

In summary, the SAP M2M Conformance Testing methodology consists of:

- Selection of Implementations Under Test (IUT).
- Identification of reference points.
- Development of test specifications, which includes:
  - Development of "Implementation Conformance Statements" (ICS), if not already provided as part of the base standard.
  - Development of "Test Suite Structure and Test Purposes" (TSS & TP).
  - Development of "Abstract Test Suite" (ATS) including:
    - Definition of the Abstract Protocol Tester (APT).
    - Definition of TTCN-3 test architecture.
    - Development of TTCN-3 test suite, e.g. naming conventions, code documentation, test case structure.

5.2 Implementation Under Test

5.2.1 Introduction

The "Implementation Under Test" (IUT) is a protocol implementation considered as an object for testing. This means that the test process will focus on verifying the compliance of this protocol implementation (IUT) with requirements set up in the related base standard. An IUT normally is implemented in a "System Under Test" (SUT). For testing, an SUT is connected to a test system over at least a single interface. Such an interface is identified as "Reference Point" (RP) in the present document. Further details on RPs are presented in clause 5.2.

NOTE: Other interfaces between the test system and the IUT may be used to control the behaviour of the IUT during the test process.

IUTs normally are entities of a protocol architecture for a specific communication protocol located in an OSI layer. Figure 5.2.1-1 shows a complete view of communication layer for M2M domain. Further details are presented in the clause 5.2.2.

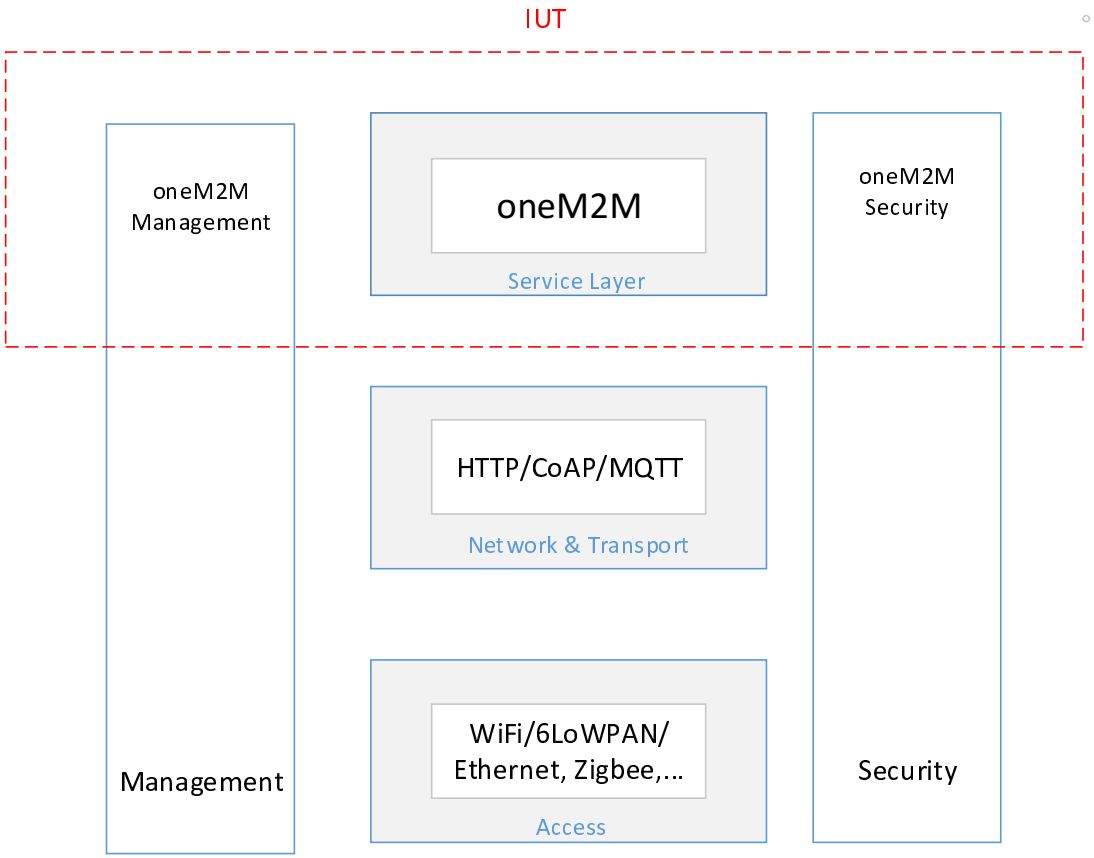

Figure 5.2.1-1: Example of IUT in the oneM2M reference architecture

5.2.2 Service Layer Communication

5.2.2.1 oneM2M implementation

Table 5.2.2.1-1 shows the IUTs for oneM2M reference architecture as defined in ETSI TS 118 101 [3].

Table 5.2.2.1-1: IUTs for oneM2M

IUT (node)	Entities	Interfaces	Notes
ASN	Application Entity (AE)	Mca	
	Common Services Entity (CSE)	Mca, Mcc, Mcn	
ADN	Application Entity (AE)	Mca	
MN	Application Entity (AE)	Mca	
	Common Services Entity (CSE)	Mca, Mcc, Mcn	
IN	Application Entity (AE)	Mca	
	Common Services Entity (CSE)	Mca, Mcc, Mcn	
Any of above	Network Services Entity (NSE)	Mcn	

5.2.2.2 SAP implementation

In Smart Appliance Testing, **only the AE is tested**.

Table 5.2.2.2-1: IUTs for Smart Appliance

IUT (node)	Entities	Interfaces	Notes
ASN	Application Entity (AE)	Mca	
ADN	Application Entity (AE)	Mca	
MN	Application Entity (AE)	Mca	
IN	Application Entity (AE)	Mca	

5.3 Identification of the Reference points

This clause illustrates candidate reference points (RPs) where test systems can be connected in order to test conformance of oneM2M protocols (IUTs) with oneM2M base standards.

Table 5.3-1: RPs for oneM2M

RP Identifier	RP Type	oneM2M node-entity	oneM2M node-entity	Network
RP-SAP-1	Mca	ASN-AE	ASN-CSE	
RP-SAP-2	Mca	MN-AE	MN-CSE	
RP-SAP-3	Mca	IN-AE	IN-CSE	
RP-SAP-4	Mca	ADN-AE	IN-CSE	
RP-SAP-5	Mca	ADN-AE	MN-CSE	

5.4 Development of Conformance Test Specifications

5.4.1 Protocol Implementation Conformance Statements (PICS)

The guidance to produce ICS pro forma is provided in ETSI EG 201 058 [i.5] and no extra guidance is required for the context of Smart Appliance.

5.4.2 Test Suite Structure & Test Purposes (TSS & TP)

5.4.2.1 Introduction

A test purpose is a prose description of a well-defined objective of testing. Applying to conformance testing, it focuses on a single conformance requirement or a set of related conformance requirements from the base standards.

Several types of presentation of the test purposes exist. These presentations are combining text with graphical presentations, mainly tables, and include sometimes message sequence charts. The present document presents a proposed table template to write test purposes with recommendations concerning the wording and the organization of the test purposes.

There are usually numerous test purposes, which need to be organized in structured groups. The organization of the test purposes in groups is named "Test Suite Structure".

The development of the test purposes follows the analysis of the conformance requirements, clearly expressed in the base standards. Furthermore, the analysis of a base standard leads to the identification of different groups of functionalities, which are used to define the first levels of the test suite structure.

The guidance provided in the following clauses is based on two ETSI reference documents produced by the MTS technical committee ETSI ETR 266 [i.6] and ETSI ETS 300 406 [i.7].

5.4.2.2 Test Suite Structure

Defining the test suite structure consists of grouping the test purposes according to different criteria like for instance:

- The functional groups and sub-groups of procedures in the base standard, from which the requirement of the test purpose is derived.
- The category of test applying to the test purposes, for instance:
  - valid behaviour test;
  - invalid behaviour test;
  - timer test;
  - etc.

Usually the identification of the different functional groups of procedures leads to the definition of the top levels of the TSS. Then further levels at the bottom of the TSS is used to group test purposes belonging to the same type of test.

Table 5.4.2.2-1 shows SAP Test Suite Structure (TSS) including its subgroups defined for conformance testing.

Table 5.4.2.2-1: TSS for SAP TST

Root	Group	Sub-group	category
SAP	SAREF	XXX	Valid behaviour
			Valid behaviour
			Invalid Syntax or Behaviour Tests
			Inopportune Behaviour

The test suite is structured as a tree with the root defined as SAP. The tree is of rank 3 with the first rank a Group, the second a Sub-group and the third a Category. The third rank is the standard ISO conformance test categories.

5.4.2.3 Test Purpose

5.4.2.3.1 Introduction

A test purpose is an informal description of the expected test behaviour. As such it is written in prose.

In order to increase the readability of the TP, the following two recommendations should be followed:

- Each TP should be presented in a table, containing two main parts:
  - The TP header, which contains the TP identifier, the TP objective and the external references (PICS, and base standard).
  - The behaviour part, which contains the test behaviour description. This part can be optionally divided in the three following parts, in order to increase the readability:
    - the initial conditions;
    - the expected behaviour;
    - the final conditions.
- The prose describing the test behaviour (including initial and final conditions) should follow some rules, as for instance the use of reserved keywords and syntax.

Table 5.4.2.3.1-1: TP pro-forma template

TP Id	
Test objective	
Reference	
Config Id	
PICS Selection	
Initial conditions	**with {** **}**
Expected behaviour	**Test events**
	when { **}**
	then { **}**

Table 5.4.2.3.1-2: Description of the fields of the TP pro-forma

TP Header	
TP ID	The TP ID is a unique identifier. It shall be specified according to the TP naming conventions defined in clause 5.4.2.3.2.
Test objective	Short description of test purpose objective according to the requirements from the base standard.
Reference	The reference indicates the sub-clauses of the reference standard specifications in which the conformance requirement is expressed.
PICS Selection	Reference to the PICS statement involved for selection of the TP. Contains a Boolean expression.
TP Behaviour	
Initial conditions	The initial conditions define in which initial state the IUT has to be to apply the actual TP. In the corresponding Test Case, when the execution of the initial condition does not succeed, it leads to the assignment of an Inconclusive verdict.
Expected behaviour (TP body)	Definition of the events, which are parts of the TP objective, and the IUT are expected to perform in order to conform to the base specification. In the corresponding Test Case, Pass or Fail verdicts can be assigned there.
Final conditions	Definition of the events that the IUT is expected to perform or shall not perform, according to the base standard and following the correct execution of the actions in the expected behaviour above. In the corresponding Test Case, the execution of the final conditions is evaluated for the assignment of the final verdict.

Defining the initial and final conditions, separately from the expected behaviour, makes the reading of the TP easier and avoid misinterpretations.

The "expected behaviour", which matches the events corresponding to the TP objective, can also be named "TP body", which is similar to the "test case body" in an abstract test suite (ATS).

5.4.2.3.2 TP identifier

The TP identifier identifies uniquely the test purposes. In order to ensure the uniqueness of the TP identifier, it follows a naming convention.

The more useful and straightforward naming convention consists of using the test suite structure, to form the first part of the TP identifier. Then the final part consists of a number to identify the TP order within a TP group.

Table 5.4.2.3.2-1 shows an example of TP naming convention applying to the TSS described in table 5.4.2.3.1-2.

The TP identifier is formed by the abbreviation "TP", followed by abbreviation representing the group of the following TSS levels, ending with a number representing the TP order. Each field of the TP identifier is separated by a "/".

Table 5.4.2.3.2-1: Example of TP naming convention SAP TST

Identifier:	**TP/<root>/<gr>/<sgr>/<x>/<nn>**		
	<root> = root		
	<gr> = group		
	<sgr> = subgroup		
	<x> = type of testing	BV	Valid Behaviour tests
		BI	Invalid Syntax or Behaviour Tests
		BO	Inopportune Behaviour
	<nn> = sequential number		01 to 99

A TP identifier, following the TP naming convention of the table could be TP/SAP/SAREF/BV/001.

The TP numbering uses two digits for presentation, and starts with 01 rather than with 00. Exceeding 99 TPs per group is not recommended. In such a case, it is rather recommended to create sub-groups, in order to keep clarity in the Test Suite Structure.

5.4.2.3.3 Test objective

The test objective clearly indicates which protocol requirement is intended to be tested in the test purpose. This part eases the understanding of the TP behaviour. This also eases the identification of the requirements, which were used as a basis for the test purpose.

It is recommended to limit the length of the test objective to one sentence.

See also the example in table 5.4.2.3.6-2.

5.4.2.3.4 Reference

In the reference row, the TP writer indicates, in which clauses of the protocol standards, the requirement are expressed. This information is critical, because it justifies the existence and the behaviour of the TP.

The reference row may refer to several clauses. When the clause containing the requirement is big (for instance, more than ½ page), it is recommended to indicate the paragraph of the clause where the requirement was identified.

The reference to the base standard actually is precise enough to enable the TP reader to identify quickly and precisely the requirement.

See also the example in table 5.4.2.3.6-2.

5.4.2.3.5 PICS selection

The PICS selection row contains a Boolean expression, made of PICS parameters. It is recommended to use PICS acronym, which clearly identify the role of the PICS, instead of using the PICS table and row references.

A mapping table is included in the TP document to link the PICS acronym with its corresponding reference in the PICS document. See the example table 5.4.2.3.5-1.

Table 5.4.2.3.5-1: Mnemonics for PICS reference

Mnemonic	PICS item
PICS_AE	A.5.1/1 [1]
PICS_CONTAINER	A.5.1/1 [1]
PICS_FLEXCONTAINER	A.5.1/1 [1]
PICS_GENERIC_IWK_SERVICE	A.5.1/1 [1]
PICS GENERIC_IWK_OP_INSTANCE	A.5.1/1 [1]
PICS_DEVICE	A.5.2/1 [1]
PICS_FUNCTION	A.5.3/1 [1]
PICS_ PROPERTY	A.5.5/1 [1]
PICS_ COMMAND	A.5.6/1 [1]
PICS_DEVICECATEGORY	A.5.7/1 [1]
PICS_STATE	A.5.8/1 [1]
PICS_TASK	A.5.9/1 [1]
PICS_UNITOFMESURE	A.5.10/1 [1]
PICS_COMMODITY	A.5.11/1 [1]
PICS_BUILDINGOBJECT	A.5.12/1 [1]
PICS_BUILDINGSPACE	A.5.13/1 [1]
PICS_PROFILE	A.5.14/1 [1]
PICS_FUNCTIONCATEGORY	A.5.15/1 [1]
PICS_OBJECTPROPERTY	A.5.16/1 [1]
PICS_DATATYPE	A.5.17/1 [1]
PICS_OPERATION	A.5.18/1 [1]
PICS_THING	A.5.19/1 [1]
PICS_ASPECT	A.5.20/1 [1]
NOTE: [x] being the bookmark to the PICS standard, in the reference clause of the TP document.	

5.4.2.3.6 TP behaviour

First of all, the following global rules apply, when writing the behaviour description:

- The behaviour description is written in an explicit, exhaustive and unambiguous manner.
- The behaviour description only refers to externally observable test events (send/receive PDUs, timer, counters, etc.) or to events or states, which can be directly or indirectly observed externally.
- All test events used in the behaviour description are part of the procedures specified in the protocol standards.
- The wording of the test events in the behaviour description is explicit, so that the ATS writers do not have to interpret the behaviour description.
- All test events in the behaviour description should result as far as possible in one ATS statement (for instance a TTCN statement).

The test behaviour is described in prose. This enables to use different ways to express similar behaviour. But using different expressions to define identical behaviours can lead to some misinterpretation of the test purposes. Also the meaning and the expected order of the test event have a clear and unique meaning for different readers.

Thus, the present document recommends to use pre-defined keywords in order to express clearly and uniquely the test behaviour.

Table 5.4.2.3.6-1 shows some recommended pre-defined keywords and their context of usage. The pre-defined keywords are also likely to be used in combination with the "{" "}"delimiters, in order to clearly delimitate their action in the test behaviour description.

Table 5.4.2.3.6-1 does not present an exhaustive list, so that additional keywords might be defined as necessary. The definition of additional keywords is included in the corresponding TSS & TP document.

Table 5.4.2.3.6-1: List of pre-defined keywords for the behaviour description

Behavioural keywords	
`with`	`with`, together with `"{" "}"` delimiters is used to express the initial conditions, which consist of a set of events, to be executed before starting with the test behaviour corresponding to the test objective. EXAMPLE: **with {** the CSE **being** in the "initial state" **and** the IUT **having** a Functionality class or subclass instance **and** the IUT **having** privileges to perform CREATE operation **}**
`ensure that`	`ensure that`, together with "{" "}" delimiters is used to define the place of the expected behaviour (TP body) or the final conditions. EXAMPLE: `ensure that` **when {** the IUT **starts** and **registers** **}**
`when/then`	`when` combined with `then` enables to define the test behaviour involving a combination of stimuli and response events. The when/then combination is used when the occurrence of an event is triggered by the realization of a previous event. EXAMPLE: `ensure that {` **when {** the IUT **starts** and **registers** **}** **then {** the IUT **sends** a valid CREATE request **containing** To **set to** address of <Flex*Container*> **and** Resource-Type **set to** <*semanticDescriptor*> *type* **and** From **set to** AE-ID **and** Content **containing** <*semanticDescriptor*> resource **containing** *descriptor* attribute **containing** instance of the Command class or subclass **containing** *rdf:about* attribute **containing** URI of the Device concatenated with the letter "*" and the class name of the Command. **}**

Table 5.4.2.3.6-2: Event keywords

Event keywords	
`the IUT`	Event in the TP is expressed from the point of view of the IUT. This avoid any misinterpretation.
`receives`	states for an event corresponding to the receipt of a message by the IUT.
`having received`	states for a condition where the IUT has received a message.
`sends`	states for an event corresponding to the sending of a message by the IUT.
`having sent`	states for a condition where the IUT has sent a message.

`from/to`	Indicates the destination or the origin of a message as necessary (interface, ...)
	EXAMPLE:
	`ensure that {`
	`  **when** { the IUT receives a valid XXX message **from** the YYY port.. }`

`on expiry of`	Indicate the expiry of a timer, being a stimulus for forthcoming event. EXAMPLE: `ensure that { **on expiry** of the Timer T1, the IUT sends a valid XXX message...`
`after expiry of`	Used to indicate that an event is expected to occur after the expiry of a timer. EXAMPLE: `ensure that { the IUT sends a valid XXX message **after expiry** of the minimum timer interval }`
`before expiry of`	Used to indicate that an event is expected to occur before the expiry of a timer. EXAMPLE: `ensure that { the IUT sends a valid XXX message **before expiry** of the maximum timer interval }`
Event attribute keywords	
`valid`	Indicates that the event sent or received is a valid message according to the protocol standard, thus: • containing all mandatory parameters, with valid field values; • containing required optional fields according to the protocol context, with valid field values.
`invalid`	Indicates that the event sent or received is an invalid message according to the protocol standard. Further details describing the invalid fields of the message is added. EXAMPLE: With { the IUT having sent an **invalid** XXX message **containing** no mandatory YYY parameter... }
`containing`	Enables to describe the content of a sent or received message.
`indicating`	Enables to specify the interpretation of the value allocated to a message parameter. EXAMPLE: With { the IUT having sent a **valid** XXX message **containing** a mandatory YYY parameter **indicating** "ZZZ supported"... }
Logical keywords	
`and`	Used to combine statements of the behaviour description.
`or`	
`not`	

Table 5.4.2.3.6-3: TP example for SAP TST

TP Id	TP/SAP/SAREF/BV/001
Test objective	Check that the IUT creates for an instantiation of a *class* of the Base Ontology a oneM2M resource of type <*semanticDescriptor*> containing a descriptor attribute containing the instantiated class in RDF data.
Reference	ETSI TS 118 112 [5], clause 7.1.1.1
Config Id	CF03
PICS Selection	PICS_AE
Initial conditions	**with {** the CSE **being** in the "initial state" **and** the IUT **having** an ontology instance **containing** an instantiation of a *class* of the Base Ontology **and** the IUT **having** privileges to perform CREATE operation **}**
Expected behaviour	**Test events**
	when { the IUT **starts** and **registers** **}**
	then { the IUT **sends** a valid CREATE request **containing** To **set to** address of <AE> resource **and** Resource-Type **set to** <*semanticDescriptor*> **and** From **set to** AE-ID **and** Content **containing** <*semanticDescriptor*> resource **containing** descriptor attribute **containing** RDF data of the instantiated class **}**

Table 5.4.2.3.6-4

TP Id	Class
TP/SAP/SAREF/BV/001_01	saref:Device
TP/SAP/SAREF/BV/001_02	saref:Door switch
TP/SAP/SAREF/BV/001_03	saref:Energy meter
TP/SAP/SAREF/BV/001_04	saref:Light switch
TP/SAP/SAREF/BV/001_05	saref:Meter
TP/SAP/SAREF/BV/001_06	saref:Sensor
TP/SAP/SAREF/BV/001_07	saref:Smoke sensor
TP/SAP/SAREF/BV/001_08	saref:Switch
TP/SAP/SAREF/BV/001_09	saref:Temperature sensor
TP/SAP/SAREF/BV/001_10	saref:Washing machine
TP/SAP/SAREF/BV/001_11	saref:Service
TP/SAP/SAREF/BV/001_12	saref:Switch on service
TP/SAP/SAREF/BV/001_13	saref:Function
TP/SAP/SAREF/BV/001_14	saref:Actuating function
TP/SAP/SAREF/BV/001_15	saref:On off function
TP/SAP/SAREF/BV/001_16	saref:Open close function
TP/SAP/SAREF/BV/001_17	saref:start stopfunction
TP/SAP/SAREF/BV/001_18	saref:Event function
TP/SAP/SAREF/BV/001_19	saref:Metering function
TP/SAP/SAREF/BV/001_20	saref:Sensing function
TP/SAP/SAREF/BV/001_21	saref:Command
TP/SAP/SAREF/BV/001_22	saref:Close command
TP/SAP/SAREF/BV/001_23	saref:Get command
TP/SAP/SAREF/BV/001_24	saref:Get current meter value command
TP/SAP/SAREF/BV/001_25	saref:Get meter data command
TP/SAP/SAREF/BV/001_26	saref:Get meter history command
TP/SAP/SAREF/BV/001_27	saref:Get sensing data command
TP/SAP/SAREF/BV/001_28	saref:Notify command
TP/SAP/SAREF/BV/001_29	saref:Off command

TP Id	Class
TP/SAP/SAREF/BV/001_30	saref:On command
TP/SAP/SAREF/BV/001_31	saref:Open command
TP/SAP/SAREF/BV/001_32	saref:Pause command
TP/SAP/SAREF/BV/001_33	saref:Set level command
TP/SAP/SAREF/BV/001_34	saref:Set absolute level command
TP/SAP/SAREF/BV/001_35	saref:Set relative level command
TP/SAP/SAREF/BV/001_36	saref:Start command
TP/SAP/SAREF/BV/001_37	saref:Step down command
TP/SAP/SAREF/BV/001_38	saref:Step up command
TP/SAP/SAREF/BV/001_39	saref:Stop command
TP/SAP/SAREF/BV/001_40	saref:Toggle command

5.4.3 Abstract Test Suite (ATS)

5.4.3.1 Abstract protocol tester

An abstract protocol tester presented in figure 5.4.3.1-1 is a process providing the test behaviour for testing an IUT. Thus it will emulate a peer IUT of the same layer/the same entity. This type of test architecture provides a situation of communication which is equivalent to real operation between real M2M systems. The M2M test system will simulate valid and invalid protocol behaviour, and will analyse the reaction of the IUT. Then the test verdict, e.g. pass or fail, will depend on the result of this analysis. Thus this type of test architecture enables to focus the test objective on the IUT behaviour only.

In order to access an IUT, the corresponding abstract protocol tester needs to use lower layers to establish a proper connection to the system under test (SUT) over a physical link (Lower layers link).

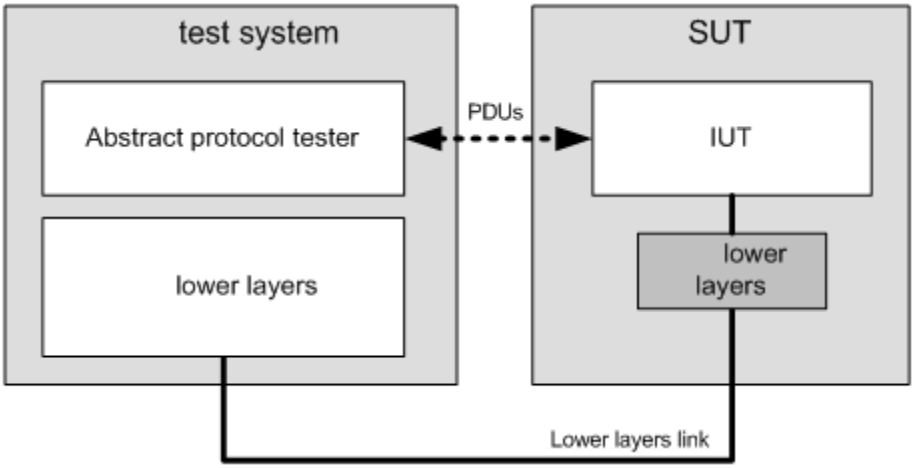

Figure 5.4.3.1-1: Generic abstract protocol tester

The "Protocol Data Units" (PDUs) are the messages exchanged between the IUT and the abstract protocol tester as specified in the base standard of the IUT. These PDUs are used to trigger the IUT and to analyse the reaction from the IUT on a trigger. Comparison of the result of the analysis with the requirements specified in the base standard allows to assign the test verdict.

Further control actions on the IUT may be necessary from inside the SUT, for instance to simulate a primitive from the upper layer or the management/security entity. Further details on such control actions are provided by means of an upper tester presented in clause 5.4.3.2.3.

The above "Abstract Test Method" (ATM) is well defined in ISO/IEC 9646-1 [i.2] and supports a wide range of approaches for testing including the TTCN-3 abstract test language ETSI ES 201 873 [i.3].

For instance, to test the oneM2M architecture, the abstract protocol tester will emulate the oneM2M primitives. In order to test the SAP on onM2M, the M2M abstract protocol tester will use e.g. HTTP or CoAP in the OSI Application Layer, TCP/UDP and IPV4/IPV6 protocol in the transport and networking layer and Ethernet technology in the access layer.

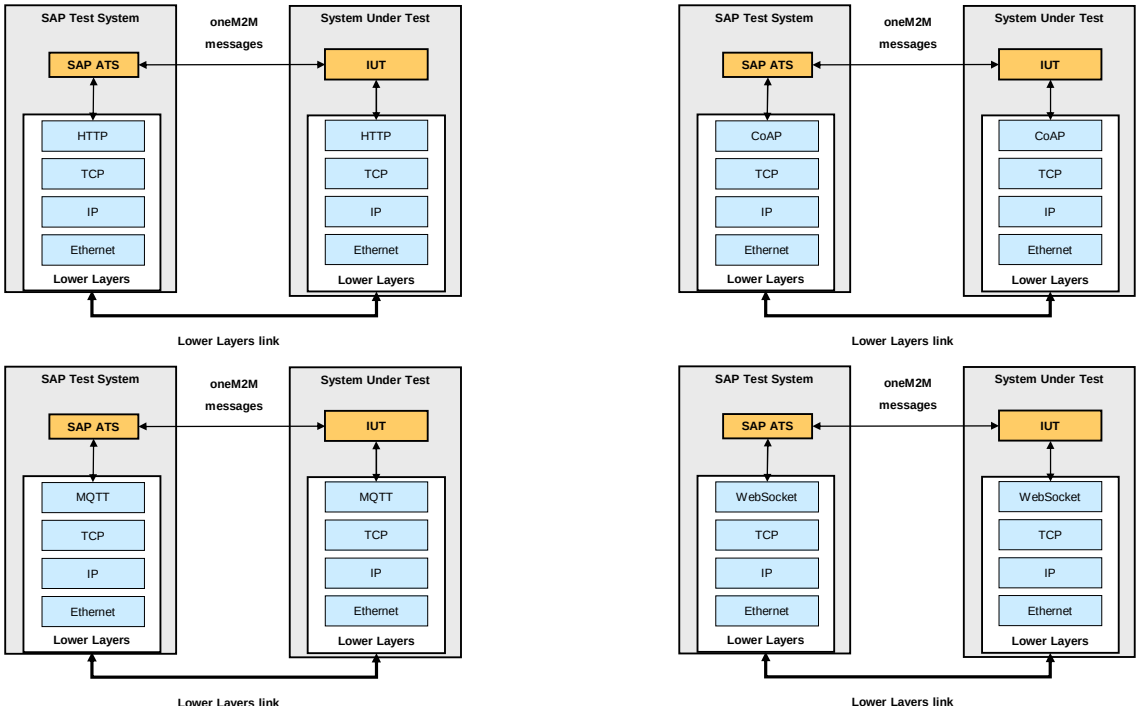

Figure 5.4.3.1-2: Abstract protocol tester for SAP on oneM2M

A current snap-shot of protocols to be tested (IUT) is shown in table 5.4.3.1-1. This table indicates which lower layer protocols (may) belong to which IUT in order to build the proper M2M test system.

Table 5.4.3.1-1: Mapping between protocols (IUTs) and lower layer protocols for Reference Point

Protocol to be tested (IUT)	Protocols of lower layers	IUT base standards
	IP, TCP, HTTP	ETSI TS 118 109 [7]
oneM2M	IP, UDP, CoAP	ETSI TS 118 108 [6]
	IP, TCP, MQTT	ETSI TS 118 110 [8]
	IP, TCP, WebSocket	ETSI TS 118 120 [9]

5.4.3.2 TTCN-3 test architecture

5.4.3.2.1 Introduction

Clause 5.4.3.2 illustrates how to implement the abstract test architecture presented in clause 5.4.3.1 in a functional test environment. There are many possibilities to implement this abstract test architecture using different types of programming languages and test devices. This SAP M2M testing framework uses TTCN-3 being a standardized testing methodology including a standardized testing language ETSI ES 201 873 [i.3], which is fully compliant with the ISO/IEC 9646 abstract test methodology [i.2].

5.4.3.2.2 Generic Conformance architecture

Figure 5.4.3.2.2-1 presents the functional test architecture for M2M conformance testing, applying the TTCN-3 test system specified in ETSI ES 201 873-5 [i.3].

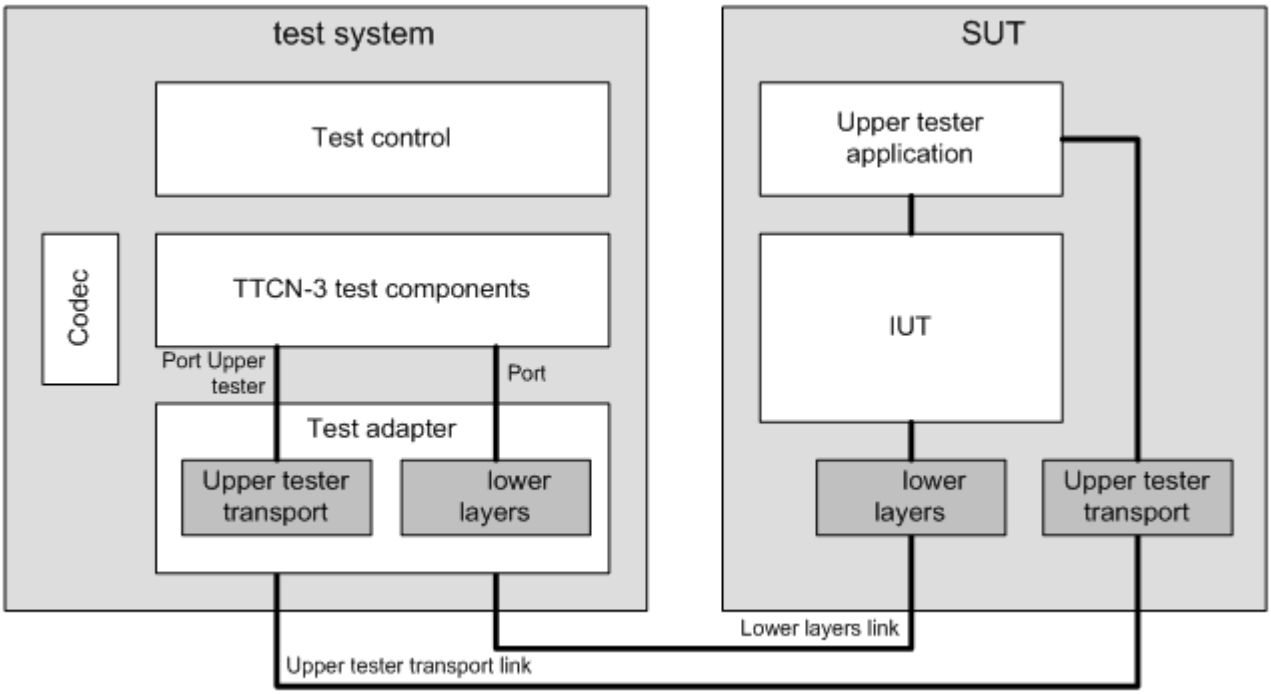

Figure 5.4.3.2.2-1: Conformance test system architecture

The "System Under Test" (SUT) contains:

- The "Implementation Under Test" (IUT), i.e. the object of the test.
- The "Upper tester application" enables to simulate sending or receiving service primitives from protocol layers above the IUT or from the management/security entity.
- The "M2M lower layers" enable to establish a proper connection to the system under test (SUT) over a physical link (Lower layers link). The lower layers link is located at a "Reference Point" (RP), see clause 5.3.
- The "Upper tester transport" is a functionality, which enables the test system to communicate with the upper tester application. Then the upper tester can be controlled by a TTCN-3 test component as part of the test process.

The "SAP test system" contains:

- The "TTCN-3 test components" are processes providing the test behaviour. The test behaviour may be provided as one single process or may require several independent processes.
- The "Codec" is a functional part of the test system to encode and decode messages between the TTCN-3 internal data representation and the format required by the related base standard.
- The "Test Control" enables the management of the TTCN-3 test execution (parameter input, logs, test selection, etc.).
- The "Test adapter" (TA) realizes the interface between the TTCN-3 ports using TTCN-3 messages, and the physical interfaces provided by the IUT.

5.4.3.2.3 Smart Appliance Conformance architecture

Figure 5.4.3.2-3-1 shows the Smart Appliances TTCN-3 test architecture design used for the Smart Appliances ATS. The Test Suite needs to interact with the System Adapter to implement the collection of TTCN-3 test cases that are intended to be used to test the Smart Appliances IUTs.

The Smart Appliances TTCN-3 test cases implement the test algorithms specified in the TSS & TP document ETSI TS 118 120 [9], including verdict logic that allows pass/fail diagnosis.

The test algorithms use the interfaces defined ETSI TS 103 264 [1] and ETSI TS 103 267 [2] (Mca, Mcc) in order to:

1) control the test event to be sent towards the IUT; and

2) observe the test events received from the IUT.

In TTCN-3 these two interfaces have been implemented through a logical TTCN-3 concept called port (mcaPort and mccPort respectively) which allows oneM2M message primitives exchange with the IUT.

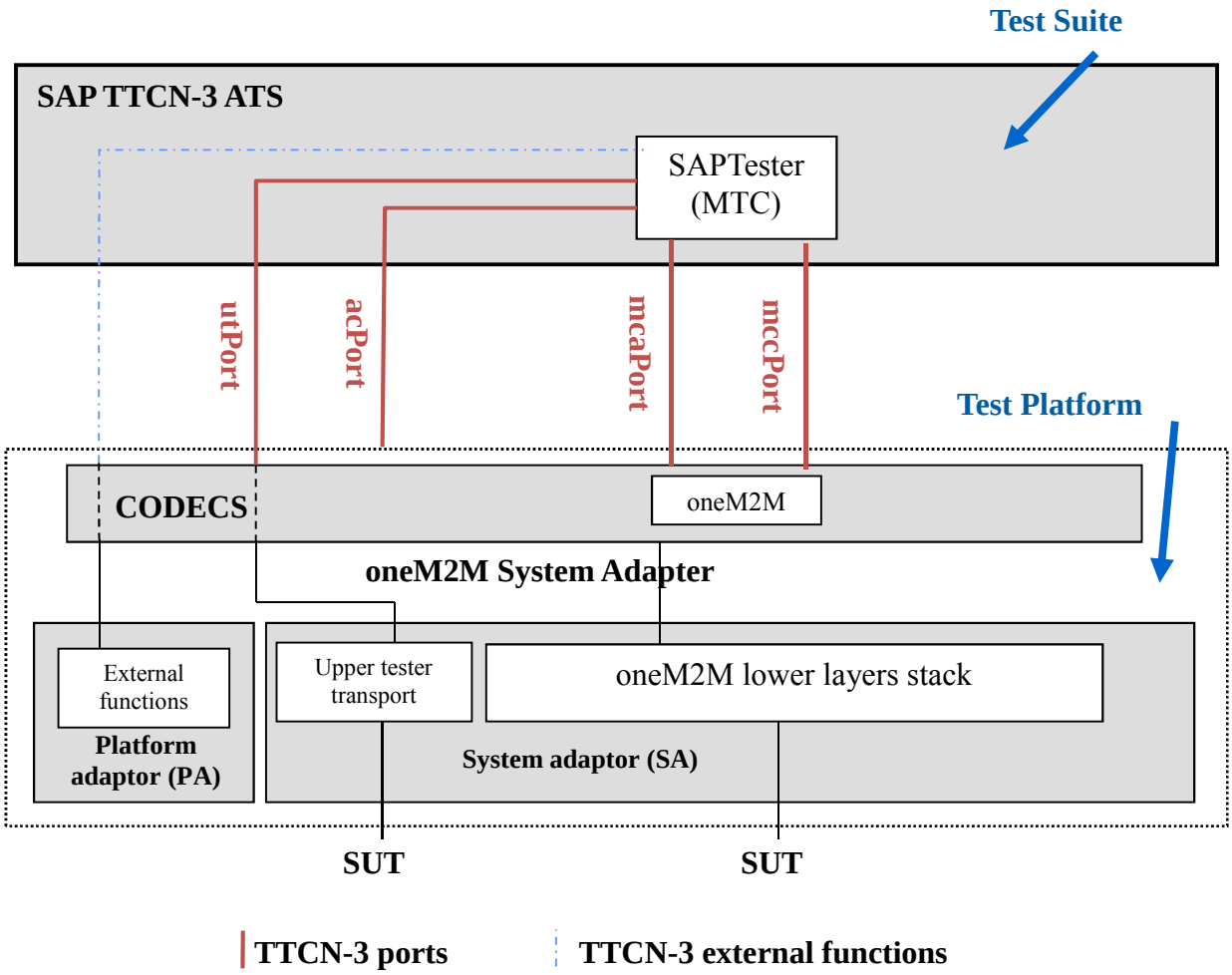

Figure 5.4.3.2.3-1: Smart Appliance Test Architecture

- **Smart Appliances TTCN-3 Abstract Test Suite:** the test suite is platform independent, and it is the cornerstone of the architecture. It allows a complete decoupling between the test suite and the rest of the test system. The test suite is composed of a complete set of test cases covering Smart Appliances requirements specified by ETSI TS 103 264 [1] and ETSI TS 103 267 [2].

- **Smart Appliances System Adapter:** this is the platform dependent part that includes adapters and codecs (out of the scope of the present document). This part of the architecture definition depends on the specific platform (e.g. Windows or Linux) and test tool on which the tester is going to run.

The oneM2M primitive messages have been mapped into TTCN-3 structure. Through this mapping, the TTCN-3 is able to build and send these messages, as well as receive them via the mcaPort and mccPort.

Additionally, the test cases are able to control and configure the test platform through a dedicated port called acPort.

To build up a tester, the test platform shall be also developed (out of scope). This test platform is composed of three adaptation layers:

PA (Platform Adapter) layer functionality implements the communication between the TTCN-3 modules and external elements that constitute the test tool such as timers and external functions. The External functions are a powerful resources supported by TTCN-3 language. An External function is a function declared at the TTCN-3 level but implemented at the native level.

SA (System Adapter) layer functionality is divided into two modules:

- oneM2M lower layers stack module implements the communication with the IUT and carries out the oneM2M primitives messages sent to or received from the IUT. This module is based on TCP or UDP depending on the binding supported by the IUT. The binding is a system adapter parameter.
- Upper Tester Transport module implements functions that enable triggering specific actions or behaviour on the IUT.

CODECS layer is the part of the tester to encode and decode messages between the TTCN-3 abstract internal data representation and the format required by the related base standard which the IUT understands. Several CODECS are required in Smart Appliances tester to cope with the bindings considered in oneM2M (HTTP, CoAP, MQTT, WebSocket) and the serialization methods (xml, json).

5.4.3.3 TTCN-3 test suite

5.4.3.3.1 TTCN-3 Test architecture

This clause presents the global M2M TTCN-3 test architecture to be used as a basis to develop further TTCN-3 test suites. This information will be used to provide the ATS documentation as part of the TTCN-3 test specification deliverables.

5.4.3.3.2 Importing XSD definition

The ETSI 118 104 [4] uses XSD for the definition of the message types. The process for using XSD data types and values in TTCN-3 modules consists of importing the existing XSD productions. For this purpose, the TTCN-3 "**`import from`**" statement should be used, in association with the "**`language`**" statement.

5.4.3.3.3 The TTCN-3 naming conventions

TTCN-3 core language contains several types of elements with different rules of usage. Applying naming conventions aims to enable the identification of the type when using specific identifiers according to the type of element.

For instance, a variable declared in a component has different scoping rules than a local variable declared in a test case. Then identifiers of component variables are different from identifiers of local variables, in order to recognize which type of variable the identifier belongs to.

Furthermore, applying naming conventions maintains the consistency of the TTCN-3 code across the test suites, and thus increase the readability for multiple users and ease the maintenance.

Table 5.4.3.3.3-1 shows the generic naming conventions applied in the ETSI test suites.

Table 5.4.3.3.3-1: ETSI TTCN-3 generic naming conventions

Language element	Naming convention	Prefix	Example identifier
Module	Use upper-case initial letter	none	SAPTemplates
Group within a module	Use upper-case initial letter	none	messageGroup
Data type	Use upper-case initial letter	none	SetupContents
Message template	Use upper-case initial letter	m_	m_setupInit
Modifying message template	Use upper-case initial letters	mw_	mw_anyUserReply
Modifying message template with wildcard	Use lower-case initial letter	md_	md_setupInit
or matching expression	Use lower-case initial letter	mdw_	mdw_anyUserReply
Signature template	Use lower-case initial letter	s_	s_callSignature
Port instance	Use lower-case initial letter	none	signallingPort
Test component instance	Use lower-case initial letter	none	userTerminal
Constant	Use lower-case initial letter	c_	c_maxRetransmission
Constant (defined within component type)	Use lower-case initial letter	cc_	cc_minDuration
External constant	Use lower-case initial letter	cx_	cx_macId
Function	Use lower-case initial letter	f_	f_authentication()
External function	Use ETSI numbering	fx_	fx_calculateLength()
Altstep (incl. Default)	Use lower-case initial letter	a_	a_receiveSetup()
Test case	Use ETSI numbering	TC_	TC_COR_0009_47_ND
Variable (local)	Use lower-case initial letter	v_	v_macId
Variable (defined within a component type)	Use lower-case initial letters	vc_	vc_systemName
Timer (local)	Use lower-case initial letter	t_	t_wait
Timer (defined within a component)	Use lower-case initial letters	tc_	tc_authMin
Module parameters for PICS	Use all upper case letters	PICS_	PICS_DOOROPEN
Module parameters for other parameters	Module parameters for other parameters	PX_	PX_TESTER_STATION_ID
PX_TESTER_STATION_ID	Use lower-case initial letter	p_	p_
Enumerated Values	Use lower-case initial letter	e_	e_syncOk

Next to such general naming conventions, table 5.4.3.3.3-2 shows specific naming conventions that apply to the SAP TTCN-3 test suite.

Table 5.4.3.3.3-2: M2M specific TTCN-3 naming conventions

Language element	Naming convention	Prefix	Example identifier
SAP module	Use upper-case initial letter	AtsSap_	AtsSap_
Module containing types and values	Use upper-case initial letter	AtsSap_TypesAndValues	AtsSap_TypesAndValues
Module containing Templates	Use upper-case initial letter	AtsSap_Templates	AtsSap_Templates
Module containing test cases	Use upper-case initial letter	AtsSap_TestCases	AtsSap_TestCases
Module containing functions	Use upper-case initial letter	AtsSap_Functions	AtsSap_Functions
Module containing external functions	Use upper-case initial letter	AtsSap_ExternalFunctions	AtsSap_ExternalFunctions
Module containing components, ports and message definitions	Use upper-case initial letter	AtsSap_Interface	AtsSap_Interface
Module containing main component definitions	Use upper-case initial letter	AtsSap_TestSystem	AtsSap_TestSystem
Module containing the control part	Use upper-case initial letter	AtsSap _TestControl	AtsSap_TestControl

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

- Structure of data:
  - Types should be defined in alphabetic order within TTCN-3 groups within the same TTCN-3 module.
  - All message types referenced in port type definitions and related to the same interface should be defined in the module, where these ports are defined.
- Log Statement:
  - All TTCN-3 log statements shall follow the following format:
    - Three asterisks should be used to precede the log text.
    - The TTCN-3 test case/function identifier in which the log statement is defined should follow.
    - One of the categories INFO, WARNING, ERROR, PASS, FAIL, INCONC, TIMEOUT should follow.
    - Free text should follow.
    - And the log text should end with three asterisks.

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

  - Any invocation of an external function shall be followed by log statement.
  - Each TTCN-3 setverdict statement that sets a test component verdict to INCONC or FAIL should be preceded by a log statement or log statement feature as first defined in TTCN-3 version 3.4.1 should be used, where the comment is part of the setverdict statement.

    ITS TTCN-3 Test suite preferably is regularly checked by the ETSI T3Q tool in order to keep the readability, consistency and maintainability of the TTCN-3 code.

5.4.3.3.4 TTCN-3 code documentation

In order to document the TTCN-3 code, the SAP TTCN-3 test suite shall follow the ETSI ES 201 873-10 [i.3].

Some of the most important rules are the following:

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

5.4.3.3.5 Test cases structure

In order to keep the readability, consistency and maintainability of the TTCN-3 code, SAP test cases are implemented following a certain structure. This framework defines this structure as follows:

- Local variables, contains the declaration of the variables to be used within the test case.
- Test control PICS, contains one or more logical expressions that use module parameters (PICS or PIXIT) in order to decide whether or not to run the test case.
- Initialization of component variables, contains zero or more assignments that initialize or modify the value of component variables.
- Test component configuration, initialized or configures the components needed to run the test.
- Test adapter configuration (optional), configures the test adapter as needed to run the test.
- Preamble, contains the actions needed to bring the IUT into the desired state.

- Test body, contains the actions needed to run the test case. This part will set the verdict of the test.
- Postamble, contains the actions needed to bring the IUT into the initial state or the desired final state.

These are the minimum sections that a test case shall contain (even being empty) but this structure may be complemented or extended with another sections as needed.

5.4.4 Protocol Implementation eXtra Information for Testing (PIXIT)

The PICS contains base specification dependent information. To derive executable tests this is insufficient; also information about the IUT and its environment shall be supplied. Such information is called Protocol Implementation eXtra Information for Testing (PIXIT).

A PIXIT pro forma identifies which ICS items are to be tested and which parameters to be instantiated for the TSS & TP being developed. The production of a PIXIT pro forma is specified in ISO/IEC 9646-6 [i.2]. A supplier, providing an IUT for conformance testing, is required to complete a IXIT pro forma, which contains additional questions that need to be answered in order to turn on/off the "switches" and identify Means of Testing for a particular Implementation Under Test (IUT).

The PIXIT may contain address information of the IUT, or parameter and timer values which are necessary for the execution of the test suite. The PIXIT information is supplied by the supplier of the IUT to the testing laboratory. To guide production of the PIXIT the testing laboratory provides a PIXIT pro forma.

The selected and implemented test cases with parameter values according to the PIXIT form the executable test suite, which are executed on a test system. The testing laboratory uses the PIXIT values stated in the PIXIT pro forma for executing test cases according to the capabilities of the Implementation Under Test. Supported values are given as a single value or a range depending on the nature of the parameter.

The PIXIT pro forma is usually provided in annex of the ATS document.

6 Interoperability testing

6.1 Introduction

Interoperability testing can demonstrate that a product will work with other like products: it proves that end-to-end functionality between (at least) two devices is as required by the standard(s) on which those devices are based. In that context, the system under test is made of the combination of different devices under test coming from different suppliers.

The important factors which characterize interoperability testing are:

- interoperability tests are performed at interfaces that offer only normal user control and observation (i.e. not at specialized interfaces introduced solely for testing purposes);
- interoperability tests are based on functionality as experienced by a user (i.e. they are not specified at the protocol level). In this context a user may be human or a software application;
- the tests are performed and observed at functional interfaces such as Man-Machine Interfaces (MMIs), protocol service interfaces and Application Programming Interfaces (APIs).

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

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

The main components of these guidelines are as follows:

- basic concepts definition;

- development of interoperability test specifications, including:
  - definition of a generic SUT architecture;
  - specification of Test configuration;
  - identification of interoperable functions;
  - development of interoperability test descriptions;
- interoperability testing process.

6.2 Basic concepts for interoperability testing

6.2.1 Overview

Figure 6.2.1-1 presents the main concepts used in the context of interoperability testing and described in clause 6.

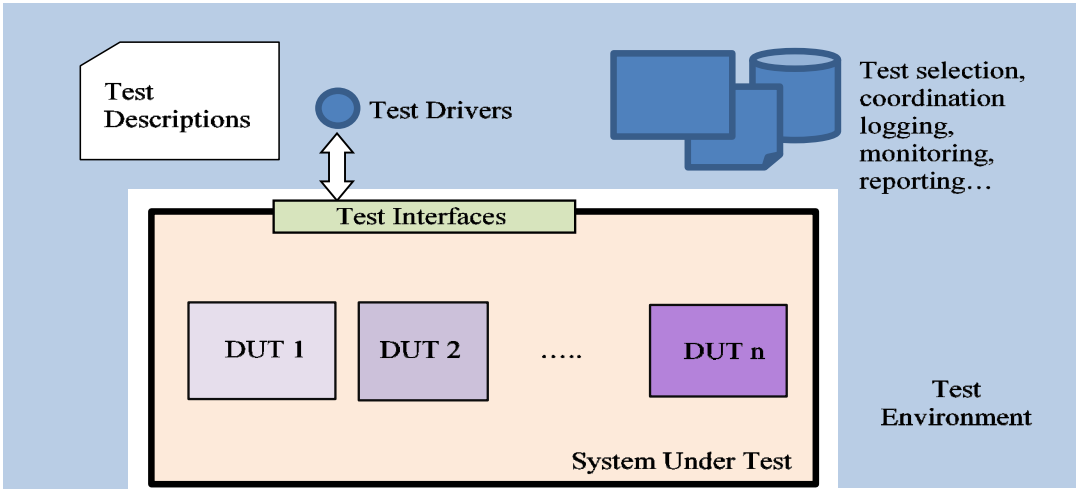

Figure 6.2.1-1: Illustration of basic concepts

6.2.2 System Under Test (SUT)

6.2.2.1 Introduction

In the context of interoperability testing, the System Under Test (SUT) is made of a number of Devices Under Test (DUTs) coming from different suppliers.

Depending on the complexity of the end-to-end system, the overall amount of DUTs under study, and the interactions among them, it might be advisable to define different SUT configuration addressing specific functional areas or groups of tests.

The first steps towards defining an Interoperability Tests Specification are identifying the Devices Under Test and describing a generic architecture where all the required SUT configurations will fit in.

6.2.2.2 Devices Under Test (DUT)

In the context of Smart Appliances, a Device Under Test is a combination of software and/or hardware items which implement the functionality of Smart Appliances and interact with other DUTs via one or more reference points.

NOTE: When using Interoperability Test Specifications in a certification scheme, the notion of Qualified Equipment (QE) or Qualified Device (QD) applies. A QD is a DUT that has successfully been tested with other QDs. The usage of interoperability Test Specifications in a certification scheme is out of the scope of the present document. Further details on this topic can be found at ISO/IEC 9646 [i.2].

6.2.2.3 Test interfaces

The interfaces that are made available by the SUT to enable the testing are usually known as the test interfaces. These interfaces are accessed by the test drivers to trigger and verify the test behaviour, Other interfaces offered by the SUT can be used for monitoring, log analysis, etc.

In the simplest case, the test interfaces will be the normal user interfaces offered by some of the DUTs (command line, GUI, web interface,…). In other cases, DUTs may offer APIs over which interoperability testing can be performed either manually using a dedicated application, or automatically using a programmable test device. In some cases, observing and verifying the functional behaviour or responses of one DUT may require to analyse its logs or records.

Additionally, while in the context of interoperability testing interfaces between the DUTs are not considered to be test interfaces, combining interoperability testing with conformance checks may require to monitor those interfaces to assess the conformance of the exchanged information or messages.

6.2.3 Test Environment

6.2.3.1 Introduction

Interoperability testing involves control and observation at the functional (rather than protocol) level. The Test Environment is the combination of equipment and procedures enabling testing the interoperability of the DUTs. Entities in the test environment access the different Devices Under Test via the Test Interfaces offered by the SUT. These entities ensure the selection, interpretation and execution of the test descriptions, coordination and synchronization of the actions on the test interfaces, and provide mechanisms for logging, reporting, monitoring and observing the interactions among the DUTs, etc.

The main entities in the test environment are described in clauses 6.2.3.2 and 6.2.3.3.

6.2.3.2 Test Descriptions

A test description provides the detailed set of instructions (or steps) that need to be followed in order to perform a test. Most often, interoperability tests are described in terms of actions that can be performed by the user(s) of the endpoint device(s).

In the case where the test is executed by a human operator, test will be described in natural language. In the case where the tests are automated, a programming or test language will be used to implement the test descriptions.

The steps in the test description can be of different nature, depending on the kind of action required: trigger a behaviour on one DUT, verify the functional response on another DUT, configure the SUT (add/remove a DUT), check a log. Each step should clearly identify the DUT and/or interface targeted by the action.

6.2.3.3 Test drivers

The test driver realizes the steps specified in a test description at one specific test interface. Testing efficiency and consistency can be improved by implementing the role of the test driver via an automatic device programmed to carry out the specified test steps. This approach may require standardized test interfaces in the DUTs.

In any given instance of testing, there may be more than one test interface over which the tests will be executed. In that case, coordination among the different test drivers and synchronization of the actions performed by them will be required. This test coordination role can be played by one of the test drivers, or by and additional entity in the test environment.

6.3 Development of Interoperability Test Specifications

6.3.1 Overview

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

- describing a generic architecture for the System Under Test;
- identifying the test architecture;
- collecting requirements in the Interoperable Features Statement (IFS);
- defining a structure for the Test Specification;
- writing a Test Descriptions (TDs) for each item in the IFS.

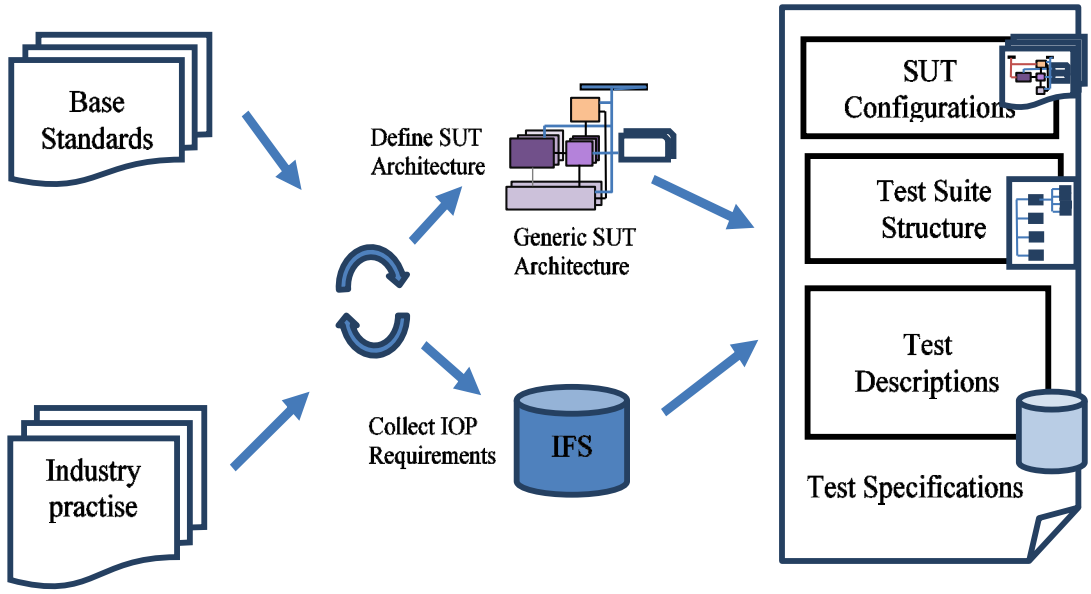

Figure 6.3.1-1: Interoperability Test Specification Development process

6.3.2 Generic SUT Architecture

A generic SUT architecture provides an abstract framework within which any specific SUT configuration should fit in. The starting point for defining a generic SUT architecture is most often the functional architecture described in the base standards, in combination with pragmatic input on how the industry and open source projects are actually implementing these functional blocks (grouping, bundling, etc.).

As described in the previous clauses, in a complex system, it may be required to define several SUT configurations to cover all the specified groups of tests. Defining the generic architecture and identifying the SUT configurations at an early stage helps to provide a structure for the test descriptions later. The generic test architecture is usually specified as a diagram and should clearly identify:

- the Devices Under Test, and the functional blocks implemented by them;
- the communications paths between the DUTs;
- if required, the protocols, APIs and/or data models to be used for communication between the DUTs.

6.3.3 Test architecture and Interfaces

A test architecture is an abstract description of logical entities as well as their interfaces and communication links involved in a test. It describes all implementation (DUTs) involved in the interoperability tests, together with the set of equipment and procedures required to enable implementations to execute the tests.

This test architecture is mainly composed of several functional entities:

- **SUT:** It is composed of a set of DUTs (M2M nodes). It is supposed that the DUTs are equipped with all the devices (sensors, etc.) needed to perform the tests.
- **Test bed control module:** This entity manages the whole test bed. It is considered to be the core of the test bed. This module synchronizes, configures, controls and runs the other entities and even the SUT. In addition, this entity gathers all the information generated by each entity in term of traces with the aim of having a global overview of the execution of the tests. Depending of the implementation of the test bed, this module might also assign the test verdicts.
- **Test stimulation environment:** This entity is in charge of stimulating the SUT for a specific test conditions.
- **Monitor:** This entity checks and gathers messages on relevant communication links.
- **SAP architecture element:** It provides SAP applications for some use cases.
- **Networks:** The test bed identifies two types of network depending on the type of information which is going to be carried out. One of the networks is used for carrying out data, and the other one is used for control.

NOTE: The definition of the test bed architecture should be done simultaneously with the test description specification.

The test bed classifies the interfaces in three groups:

- **Data:** this group contains the interfaces where data is exchanged. Depending on the type of data being exchanged, the interfaces are classified into three categories:
  - **Stimulating:** this interface carries information generated by the test bed in order to stimulate the DUTs for a specific behaviour.
  - **Monitoring:** this interface carries the protocol message exchanged between the DUTs during the execution of the tests.
  - **Tracing:** this interface carries information about the status of the execution of the DUTs and the test bed entities in order to be able to analyse as much as possible the execution of a test.
- **Control:** this group is used to configure and control the various entities in the test bed, and even the DUTs, by passing necessary parameters.
- **Test Operator:** this group provides the capability of controlling the test bed control module. Through this interface, a test operator would be able to select the test to be executed, to configure the different entities involved in the tests and to analyse the results obtained during the test execution.

Figure 6.3.3-1 illustrates interfaces involved in the test bed.

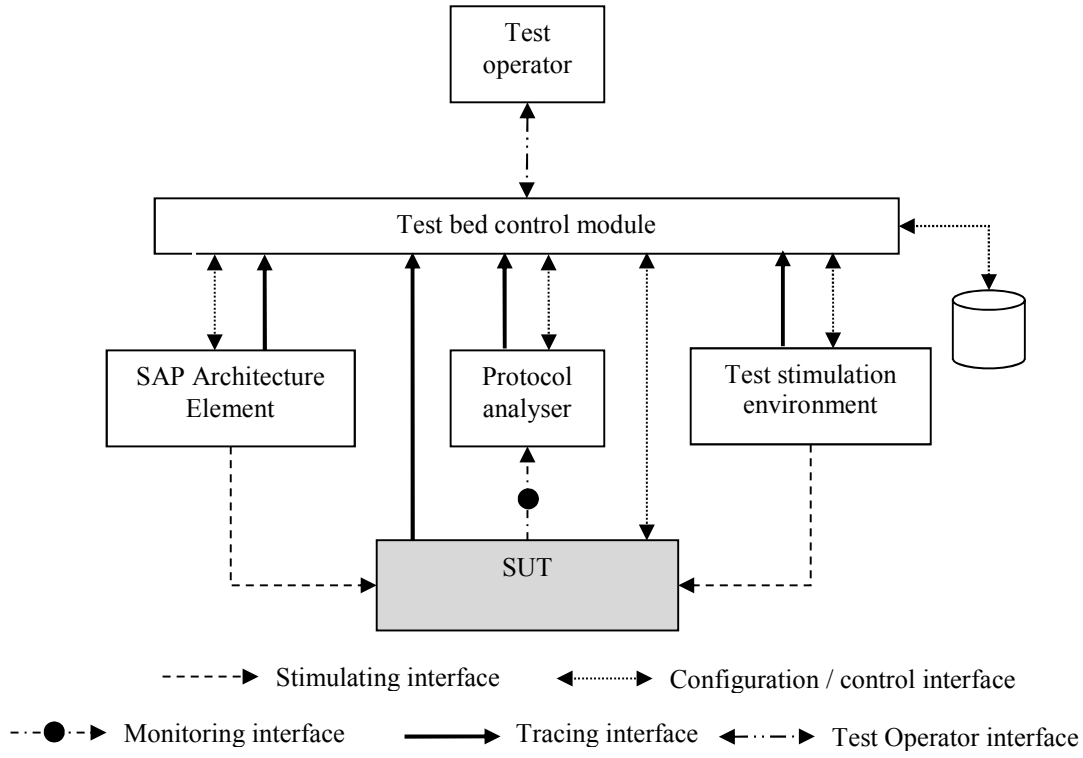

Figure 6.3.3-1: Interfaces of a test architecture

6.3.4 Interoperable Functions Statement (IFS)

An "Interoperable Functions Statement" (IFS) identifies standardized functions that a DUT shall support. These functions are either mandatory, optional or conditional (depending on other functions).

In addition, the IFS can be used as a pro forma by a manufacturer to identify the functions a DUT will support when interoperating with corresponding equipment from other manufacturers.

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

The guidance to produce IFS pro forma is provided in ETSI EG 202 237 [i.4] and no extra guidance is required for the context of SAP.

6.3.5 Test Descriptions (TD)

A "Test Description" (TD) is a well detailed description of a process that pretends to test one or more functionalities of an implementation. Applying to interoperability testing, these testing objectives address the interoperable functionalities between two or more vendor implementations.

In order to ensure the correct execution of an interoperability test, the following information should be provided by the test description:

- The proper configuration of the vendor implementations.
- The availability of additional equipment (protocol monitors, functional equipment,…) requires to achieve the correct behaviour of the vendor implementations.
- The correct initial conditions.
- The correct sequence of the test events and test results.

TDs are based on the test scenarios.

In order to facilitate the specification of test cases an interoperability test description should include as a minimum the items of the table 6.3.5-1.

Table 6.3.5-1: Interoperability test description

Identifier	a unique test description ID
Summary	a concise summary of the test which should reflect the purpose of the test and enable readers to easily distinguish this test from any other test in the document
References	a list of references to the base specification section(s), use case(s), requirement(s), TP(s) which are either used in the test or define the functionality being tested
Applicability	a list of features and capabilities which are required to be supported by the SUT in order to execute this test (e.g. if this list contains an optional feature to be supported, then the test is optional)
Configuration or Architecture	a list of all required equipment for testing and possibly also including a (reference to) an illustration of a test architecture or test configuration
Pre-Test Conditions	a list of test specific pre-conditions that need to be met by the SUT including information about equipment configuration, i.e. precise description of the initial state of the SUT required to start executing the test sequence
Test Sequence	an ordered list of equipment operation and observations. In case of a conformance test description the test sequence contains also the conformance checks as part of the observations

The TDs play a similar role as TPs for conformance testing.

The Test sequence may contain the following types of events:

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

Annex A (informative): Example of ICS table

A.1 SAREF Device Class Statement

This clause presents a list of Device classes specified in SAREF. Table A.1-1 shown as below can be used to check whether the IUT supports the listed Classes.

Table A.1-1: SAREF Device classes

Item	Resource Type	Reference	Condition	Support
1	saref:Device	[1]	M	O Yes O No
2	saref:Door switch	[1]	O.1	O Yes O No
3	saref:Energy meter	[1]	O.1	O Yes O No
4	saref:Light switch	[1]	O.1	O Yes O No
5	saref:Meter	[1]	O.1	O Yes O No
6	saref:Sensor	[1]	O.1	O Yes O No
7	saref:Smoke sensor	[1]	O.1	O Yes O No
8	saref:Switch	[1]	O.1	O Yes O No
9	saref:Temperature sensor	[1]	O.1	O Yes O No
10	saref:Washing machine	[1]	O.1	O Yes O No
O.1: at least one shall be supported.				

History

Document history		
V1.1.1	April 2017	Publication