1 Scope

The present document specifies a testing framework defining a methodology for development of conformance and interoperability test strategies, test systems and the resulting test specifications for oneM2M standards.

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 reference document (including any amendments) applies.

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

[1] oneM2M TS-0001: "Functional Architecture".

[2] oneM2M TS-0004: "Service layer Core Protocol".

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 reference document (including any amendments) applies.

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] oneM2M Drafting Rules.

NOTE: Available at http://www.onem2m.org/images/files/oneM2M-Drafting-Rules.pdf.

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

[i.3] ETSI EG 202 237: "Methods for Testing and Specification (MTS); Internet Protocol Testing (IPT); Generic approach to interoperability testing".

[i.4] 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".

3 Definitions and abbreviations

3.1 Definitions

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

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

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

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

ICS proforma: document, in the form of a questionnaire, which when completed for an implementation or system becomes an ICS

Implementation Conformance Statement (ICS): statement made by the supplier of an implementation or system claimed to conform to a given specification, stating which capabilities have been implemented

MMI	Man-Machine Interface
MQTT	Message Queue Telemetry Transport
PICS	Protocol Implementation Conformance Statement
QE	Qualified Equipment
SUT	System Under Test
TC	Test Case
TCP	Transmission Control Protocol
TD	Test Description
TP	Test Purpose
TSS	Test Suite Structure
TTCN-3	Testing and Test Control Notation version 3
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
XML	eXtensible Markup Language

4 Conventions

The key words "Shall", "Shall not", "May", "Need not", "Should", "Should not" in the present document are to be interpreted as described in the oneM2M Drafting Rules [i.1].

5 Introduction to the oneM2M 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 it to be tested".
- Definition of the procedure for development of test specifications and deliverables (for instance: TSS&TP, TP proforma, TTCN-3 test suite and documentation).

Figure 1 illustrates the oneM2M testing framework and the interactions with oneM2M base standards and test specifications. The oneM2M testing framework is based on concepts defined in ISO/IEC 9646 [i.2], TTCN-3 [i.4], ETSI EG 202 237 [i.3].

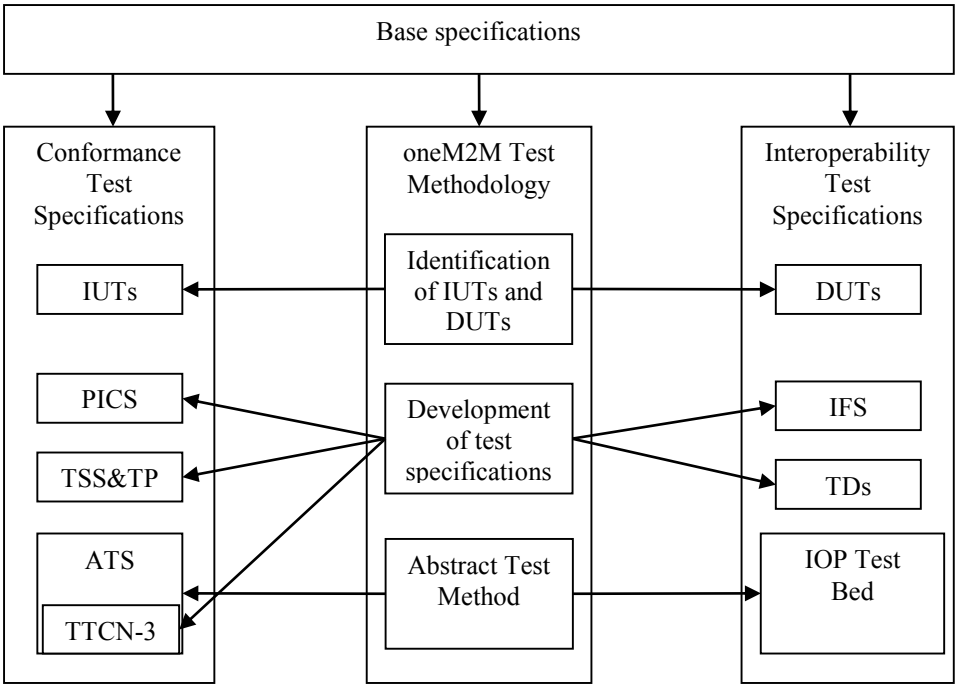


Figure 5-1: oneM2M testing methodology interactions

The 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, oneM2M 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 "oneM2M testing framework".

As its name implies, the framework is oriented towards the production of Test specifications. The oneM2M testing Framework comprises:

- a documentation structure:
  - catalogue of capabilities/features/functions (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).

6 Conformance testing

6.1 Introduction

The clause 6 shows how to apply the oneM2M conformance testing methodology in order to properly produce oneM2M 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.

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.

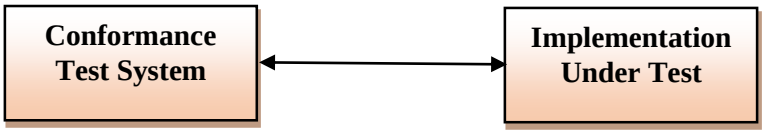


Figure 6.1-1: Conformance testing

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 features and options to be tested into a tabular form which is normally referred to as the " Implementation Conformance Statement" (ICS). All implemented capabilities supported by the Implementation Under Test (IUT) are listed by the implementer in the ICS, so that the tester knows which options have to be tested. This ensures that complete coverage is obtained.

The next step is to collect the requirements from the specification that is tested. For each requirement, 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. The TCs in the ATS are then "Verified" against a number of IUTs for correct operation according to some agreed procedures, before being released for use by the industry. An Implementation eXtra Information for Test (IXIT) proforma associated to the ATS, should be produced in supplement of the ICS document and Test Cases to help to execute Protocol conformance testing using oneM2M dedicated test equipment.

In summary, the oneM2M 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 and Implementation eXtra Information for Test" (ATS&IXIT) 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.
- Verification of ATS (TTCN-3)
- IXIT proforma.

6.2 Test architecture

6.2.1 Selection of Implementation Under Test

6.2.1.1 Definition

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, a 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 6.2.2.

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

Figure 6.2.1.1-1 shows a complete view of communication layer for oneM2M domain. Further details are presented in the following clauses.

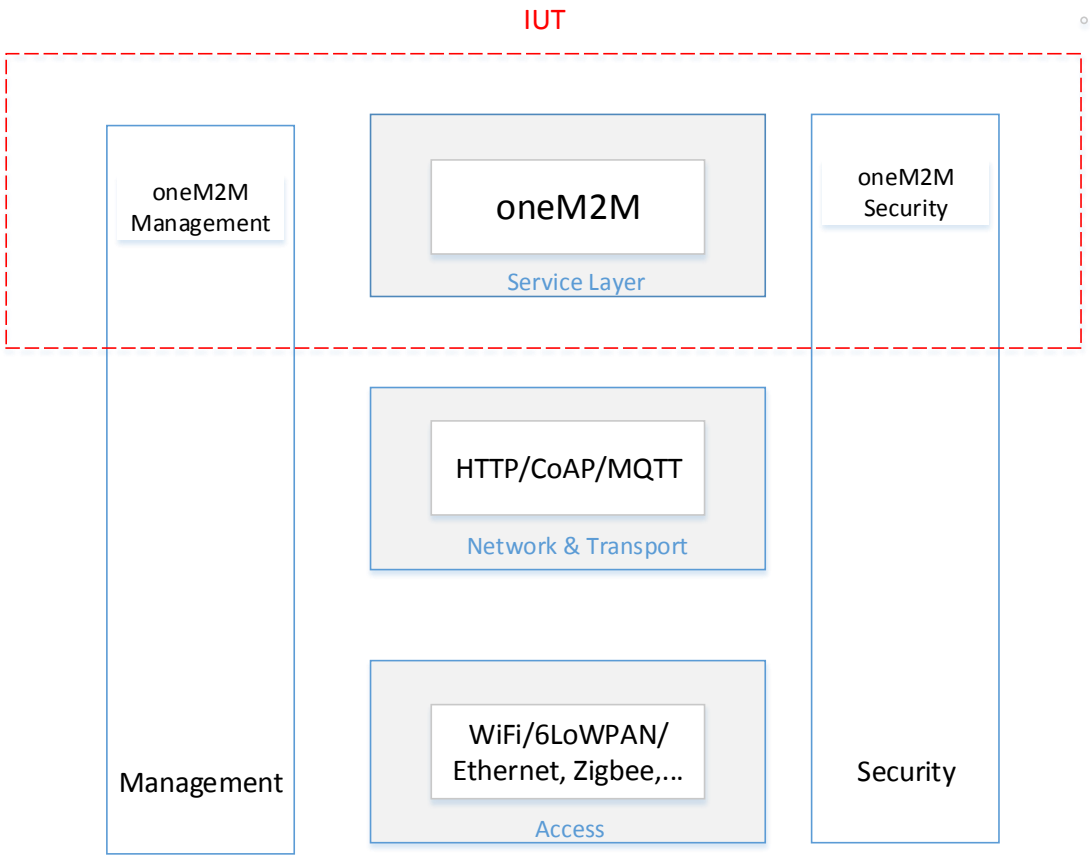


Figure 6.2.1.1-1: Example of IUT in the oneM2M reference architecture

6.2.1.2 oneM2M Service Layer Communication

Table 6.2.1.2-1 shows the IUTs for oneM2M reference architecture as defined in [1].

Table 6.2.1.2-1: 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, Mcc', Mch	
ASN/MN/IN	Network Services Entity (NSE)	Mcn	

Table 6.2.1.2-1 needs to be amended in the following cases:

- A new node or entity is defined on the base specifications.
- A new interface is defined on the base specifications between any of the existing nodes or entities.

6.2.2 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 6.2.2-1: RPs for oneM2M

RP Identifier	RP Type	oneM2M node-entity	oneM2M node-entity	Network
RP-oneM2M-1	Mca	ASN-AE	ASN-CSE	
RP-oneM2M-2	Mca	MN-AE	MN-CSE	
RP-oneM2M-3	Mca	IN-AE	IN-CSE	
RP-oneM2M-4	Mca	ADN-AE	IN-CSE	
RP-oneM2M-5	Mca	ADN-AE	MN-CSE	
RP-oneM2M-6	Mcc	ASN-CSE	IN-CSE	
RP-oneM2M-7	Mcc	ASN-CSE	MN-CSE	
RP-oneM2M-8	Mcc	MN-CSE	MN-CSE	
RP-oneM2M-9	Mcc	MN-CSE	IN-CSE	
RP-oneM2M-10	Mcn	ASN-CSE	NSE	
RP-oneM2M-11	Mcn	MN-CSE	NSE	
RP-oneM2M-12	Mcn	IN-CSE	NSE	
RP-oneM2M-13	Mcc'	IN-CSE	IN-CSE'	
RP-oneM2M-14	Mch	IN-CSE	Charging Server	

6.3 Development of Conformance Test Specifications

6.3.1 Implementation Conformance Statement (ICS)

The purpose of an ICS is to identify those standardized functions which an IUT shall support, those which are optional and those which are conditional on the presence of other functions. It helps to provide a means for selection of the suite of tests which will subsequently be developed.

In addition, the ICS can be used as a proforma for identifying which functions an IUT will support when performing conformance testing. The purpose of this ICS proforma is to provide a mechanism whereby an oneM2M implementation supplier may provide information about the implementation in a standardized manner. The information in a ICS is usually presented in tabular form as recommended in ISO/IEC 9646-7 [i.2].

The ICS can be considered as a set of "switches" which specify the capability of supporting the requirement in base standards to be tested. It is possible that with different choices in a ICS proforma, several different set of TPs will be necessary.

The ICS proforma is subdivided into clauses for the following categories of information:

- guidance for completing the ICS proforma;
- identification of the implementation;
- identification of the <reference specification type>;
- global statement of conformance

Part of an example ICS table can be found in Annex A.1.

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

6.3.2.1 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.

6.3.2.2 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 6.3.2.2-1 shows an example of a two level TSS used in the TSS&TP for the oneM2M system.

Table 6.3.2.2-1: Example of test suite structure for oneM2M system

TP/<root>/<gr>/<sgr>/<xx>/<nnn>		
<root> = root	oneM2M	oneM2M
<gr> = group	AE	Application Entity
	CSE	Common Services Entity
<sgr> = sub- group	REG	Registration
	DMR	Data Management and Repository
	SUB	Subscription and Notification
	GMG	Group Management
	DIS	Discovery
	LOC	Location
	DMG	Device Management
	CMDH	Communication Management and Delivery Handling
	SEC	Security
<xx> = type of testing	BI	Invalid Behaviour tests
	BO	Inopportune Behaviour tests
	BV	Valid Behaviour tests
<nnn> = sequential number		001 to 999

6.3.2.3 Test Purpose

6.3.2.3.1 Introduction

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

When needed to clarify the TP, it is helpful to add some graphical presentations, mainly tables, and include message sequence charts.

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 (ICS, 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 6.3.2.3.1-1: TP pro-forma template

TP Id		
Test objective		
Reference		
Config Id		
PICS Selection		
Initial conditions		
Expected behaviour	**Test events**	**Direction**
	when { **}**	IUT ← AE
	then { **}**	IUT → AE

Table 6.3.2.3.1-2: 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 the above clause.
Test objective	Short description of test purpose objective according to the requirements from the base standard.
Reference	The reference indicates the clauses of the reference standard specifications in which the conformance requirement is expressed.
ICS selection	Reference to the ICS statement involved for selection of the TP. Contains a Boolean expression.
TP Behaviour	
Initial conditions	The initial conditions defines 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).

6.3.2.3.2 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 6.3.2.3.2-1 shows an example of TP naming convention applying to the TSS described in 6.3.2.2-1.

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 6.3.2.3.2-1: Example of TP naming convention for oneM2M

TP/<root>/<gr>/<sgr>/<xx>/<nnn>		
<root> = root	oneM2M	oneM2M
<gr> = group	AE	Application Entity
	CSE	Common Services Entity
<sgr> = sub- group	REG	Registration
	DMR	Data Management and Repository
	SUB	Subscription and Notification
	GMG	Group Management
	DIS	Discovery
	LOC	Location
	DMG	Device Management
	CMDH	Communication Management and Delivery Handling
	SEC	Security
<xx> = type of testing	BI	Invalid Behaviour tests
	BO	Inopportune Behaviour tests
	BV	Valid Behaviour tests
<nnn> = sequential number		001 to 999

A TP identifier, following the TP naming convention of the table could be for instance TP/oneM2M/CSE/DMR/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.

6.3.2.3.3 Test objective

The test objective clearly indicates which 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 6.3.2.3.6-2.

6.3.2.3.4 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 6.3.2.3.6-2.

6.3.2.3.5 ICS selection

The ICS selection row contains a Boolean expression, made of ICS parameters. It is recommended to use ICS acronym, which clearly identify the role of the ICS.

A mapping table is included in the TP document to link the ICS acronym with its corresponding reference in the ICS document.

Table 6.3.2.3.5-1: Example of pre-defined keywords for ICS

Mnemonic	ICS item
PICS_REGISTRATION	A.5.2. 1/1 [ICS document]
PICS_DATA_MGMT	A.5.2. 2/2 [ICS document]
PICS_AE	A.2/1 [ICS document]
PICS_CSE	A.2/2 [ICS document]
PICS_ASN	A.1/1 [ICS document]
PICS_ADN	A.1/2 [ICS document]
PICS_IN	A.1/3 [ICS document]

6.3.2.3.6 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 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 6.3.2.3.6-1 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 6.3.2.3.6-1 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 6.3.2.3.6-1: 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 IUT having sent a container create request message and ... }`

`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 receives a valid container create request message... }`

`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 {`
	`  a XXX signal is activated }`
	`  then {`
	`  the IUT sends a message containing YYY Value indicating "True"}`
	`  }`

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 a 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 6.3.2.3.6-2: TP example for oneM2M

TP Id	TP/oneM2M/CSE/DMR/RET/BO/002	
Test objective	Check that the IUT responds with an error when the AE tries to retrieve the resource TARGET_RESOURCE_ADDRESS which does not exist	
Reference	TS-0001 10.1.2 - item 13)	
Config Id	CF01	
PICS Selection	PICS_CSE	
Initial conditions	**with {** the IUT **being** in the "initial state" **and** the IUT **having registered** the AE **and** the IUT **not having created** a resource *TARGET_RESOURCE_ADDRESS* **}**	
Expected behaviour	**Test events**	**Direction**
	when { the IUT **receives** a valid RETRIEVE request **from** AE **containing** To **set to** TARGET_RESOURCE_ADDRESS **and** From **set to** AE_ID **and** **no** Content attribute **}**	IUT ← AE
	then { the IUT **sends** a Response message **containing** Response Status Code **set to** 4004 (NOT_FOUND) **}**	IUT → AE

6.3.3 Abstract Test Suite (ATS)

6.3.3.1 Abstract protocol tester

An abstract protocol tester presented in figure 6.3.3.1-1 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 oneM2M systems. The oneM2M 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).

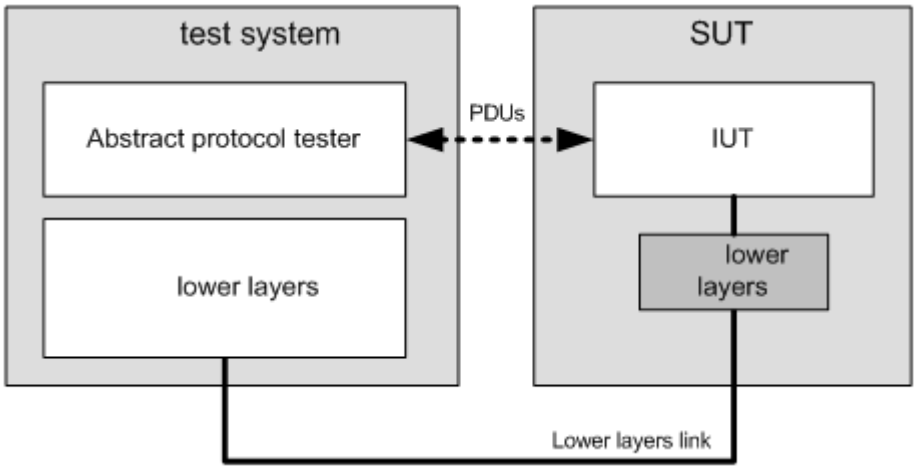


Figure 6.3.3.1-1: Generic abstract protocol tester

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 6.3.2.

The above "Abstract Test Method" (ATM) is defined in ISO/IEC 9646-1 [i.2] and supports a wide range of approaches for testing including the TTCN-3 abstract test language [i.4].

For instance, to test the oneM2M IUT, the abstract protocol tester will emulate the oneM2M primitives. use e.g HTTP, CoAP or MQTT in the OSI Application Layer, TCP/UDP and IPV4/IPV6 protocol in the transport and networking layer and Ethernet/WiFi technology in the access layer.

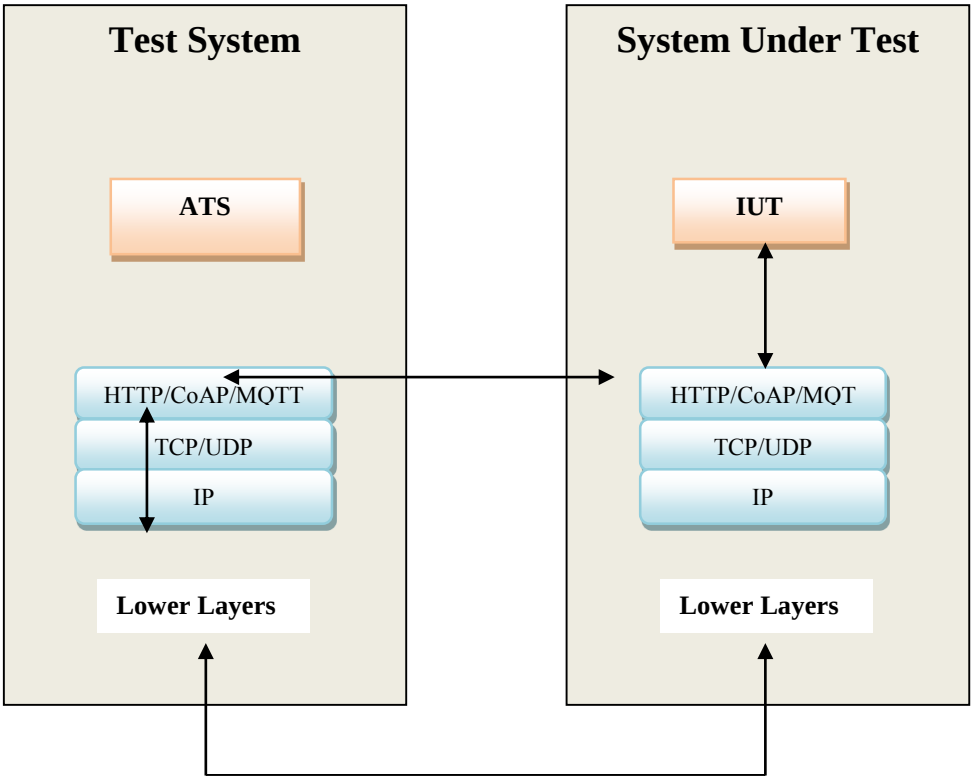


Figure 6.3.3.1-2: Abstract protocol tester for oneM2M

A current snap-shot of protocols to be tested (IUT) is shown in table 6.3.3.1-1. Table 6.3.3.1-1 indicates which lower layer protocols (may) belong to which IUT in order to build the proper M2M test system.

Table 6.3.3.1-1: Mapping between protocols (IUTs) and lower layer protocols for Reference Point

Protocol to be tested (IUT)	Protocols of lower layers	IUT base standards
oneM2M	IP, UDP, CoAP	TS-0008
	IP, TCP, HTTP	TS-0009
	IP, TCP, MQTT	TS-0010

6.3.3.2 TTCN-3 test architecture

This clause illustrates how to implement the abstract test architecture presented in clause 6.3.3.1 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 oneM2M testing framework uses TTCN-3 being a standardized testing methodology including a standardized testing language [i.4], which is fully compliant with the ISO/IEC 9646 [i.2] abstract test methodology.

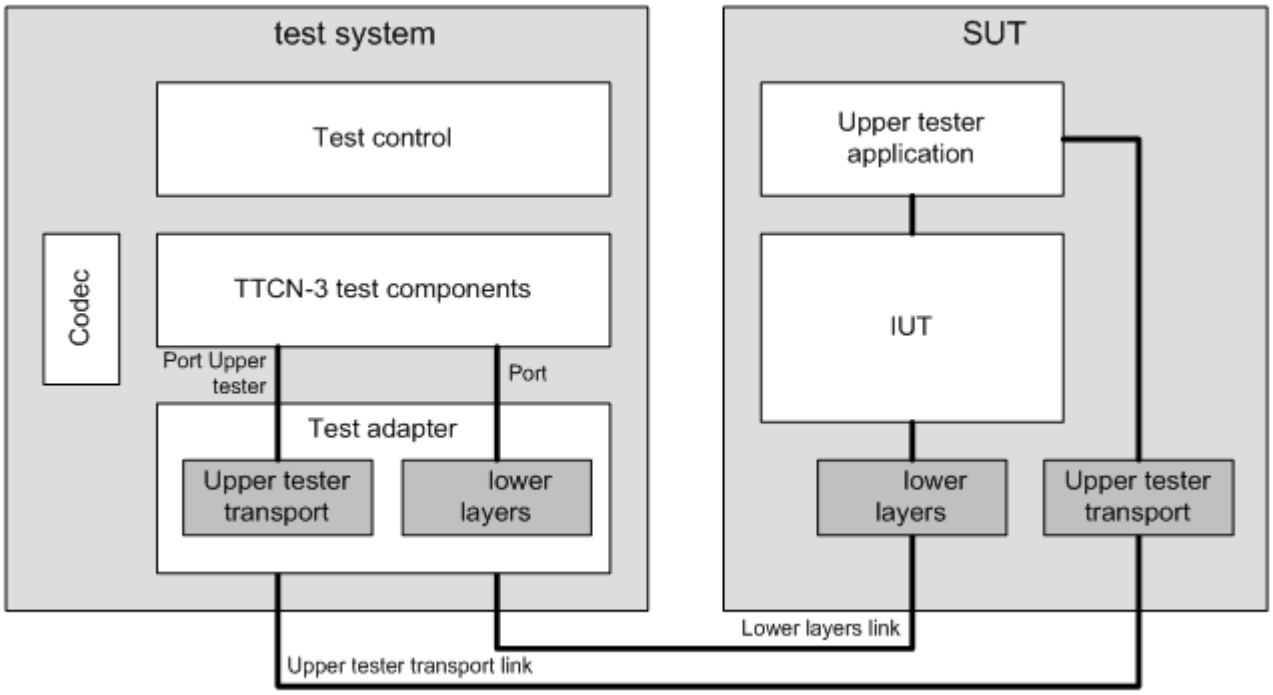


Figure 6.3.3.2-1: Conformance test system architecture

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 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 6.2.
- 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 "test system" contains:

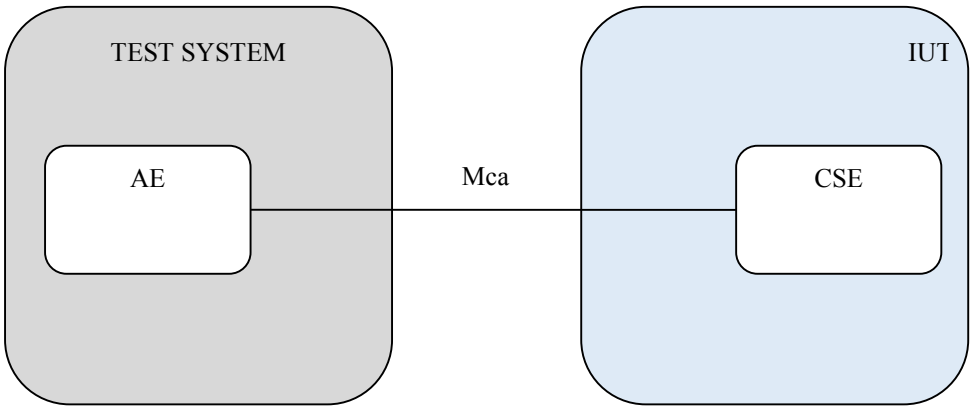
- The "TTCN-3 test components" are processes providing the test behaviour. The test behaviour may be provided as one single process or may require several independent processes.
- The "Codec" is a functional part of the test system to encode and decode messages between the TTCN-3 internal data representation 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.

6.3.3.3 Test configurations

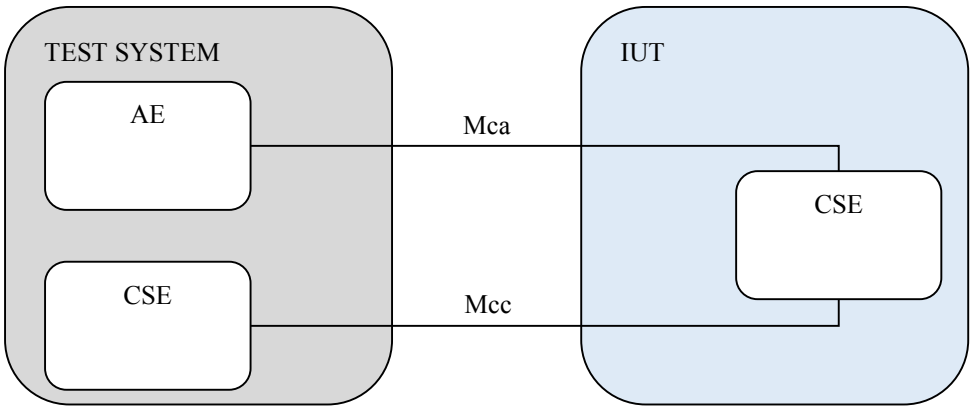
The test suite uses test configurations in order to cover the different test scenarios.

In following 2 examples, the IUT is tested by the test system simulating an AE in CF01 (no hop configuration) or an AE and a CSE in a CF02 (single hop configuration).

EXAMPLE 1: Test configuration 1 (CF01):



EXAMPLE 2: Test configuration 2 (CF02):



6.3.3.4 ATS conventions

6.3.3.4.1 Importing XSD definition

The oneM2M set of standards 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.

6.3.3.4.2 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 6.3.3.4.2-1

Language element	Naming convention	Prefix	Example identifier
Module	Use upper-case initial letter	none	OneM2M_Templates
Group within a module	Use lower-case initial letter	none	messageGroup
Data type	Use upper-case initial letter	none	SetupContents
Message template	Use lower-case initial letter	m_	m_setupInit
Message template with wildcard or matching expression	Use lower-case initial letters	mw_	mw_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 lower-case initial letter	fx_	fx_calculateLength()
Altstep (incl. Default)	Use lower-case initial letter	a_	a_receiveSetup()
Test case	Use a naming convention	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	Use all upper case letters	PX_	PX_TESTER_STATION_ID
Formal Parameters	Use lower-case initial letter	p_	p_macId
Enumerated Values	Use lower-case initial letter	e_	e_syncOk

6.3.3.5 Verification of TTCN-3

Before release for use by industry and external organisations (for example Certification Bodies) the TTCN-3 should be Verified for correct operation against a number of IUTs.

A list of all TTCN-3 test cases and their Verification status is maintained in the associated ATS. An example table to be used to record this status is given in Table 6.3.3.5-1.

Table 6.3.3.5-1: Example table for TTCN-3 Test Case Verification Status

TTCN-3 Test Case	Verification Status	TTCN-3 version used for Verification	Binding(s) used duringVerification (for information only)
TP/oneM2M/CSE/DMR/CRE/BV/004			
TP/oneM2M/CSE/DMR/CRE/BV/002	Verified	V1.3.4	HTTP, CoAP
TP/oneM2M/CSE/DMR/CRE/BV/003			

6.3.4 Implementation eXtra Information for Testing (IXIT)

The ICS 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 Implementation eXtra Information for Testing (IXIT).

An IXIT proforma identifies which ICS items are to be tested and which parameters to be instantiated for the TSS&TP being developed. The production of a IXIT Proforma is specified in ISO/IEC 9646-6 [i.2]. A supplier, providing an IUT for conformance testing, is required to complete a IXIT proforma, 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 IXIT may contain address information of the IUT, or parameter and timer values which are necessary for the execution of the test suite. The IXIT information , is supplied by the supplier of the IUT to the testing laboratory. To guide production of the IXIT the testing laboratory provides an IXIT proforma.

The selected and implemented test cases with parameter values according to the IXIT form the executable test suite, which are executed on a test system. The testing laboratory uses the IXIT values stated in the IXIT proforma 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.

7 Interoperability testing

7.1 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 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;
  - definition of Test bed architecture;
  - specification of Test scenarios and configurations;
  - identification of interoperable functions;
  - development of interoperability test descriptions;
- interoperability testing process.

7.2 Basic Concepts

7.2.1 Overview

Interoperability testing consists simply in inter-operating different vendor implementations, which are supposed to be inter-operable according to the expected conformance with the base standards. Even if this process looks easy, it requires specifying a complete environment enabling to operate vendors implementation as in real conditions. The

complete set of all vendors implementation involved in interoperability tests, together with the set of equipment required to enable vendors implementations to execute the test process is named the "Test Bed".

There are a number of different terms and concepts that can be used when describing a test methodology. The following sections describe the most important concepts used by these guidelines, which can been categorized either as part of the System Under Test (SUT) or as part of the Test Environment.

Figure 7.2.1-1 presents the main concepts used in the context of interoperability testing and described in the following sec

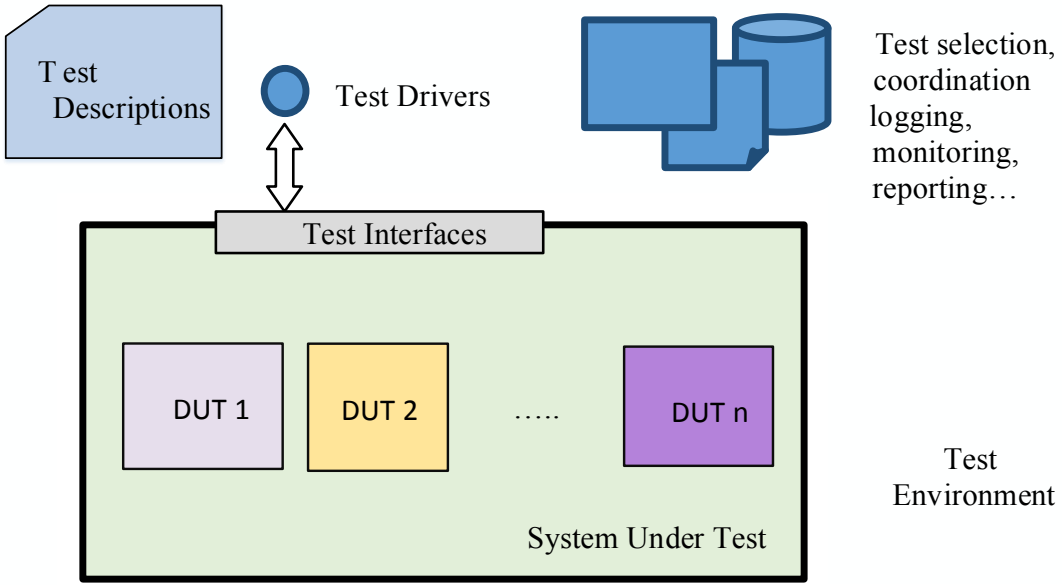


Figure 7.2.1-1: Illustration of basic concepts

7.2.2 System Under Test (SUT)

7.2.2.0 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.

7.2.2.1 Devices Under Test (DUT)

In the context of oneM2M, a Device Under Test is a combination of software and/or hardware items which implement the functionality of oneM2M 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 this document. Further details on this topic can be found at [i.3].

7.2.2.2 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, etc.). 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.

7.2.3 Test Environment

7.2.3.0 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 the following sections.

7.2.3.1 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, etc.. Each step should clearly identify the DUT and/or interface targeted by the action.

7.2.3.2 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.

7.3 Development of Interoperability Test Specifications

7.3.1 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;
- defining test scenarios;
- identifying the test bed 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.

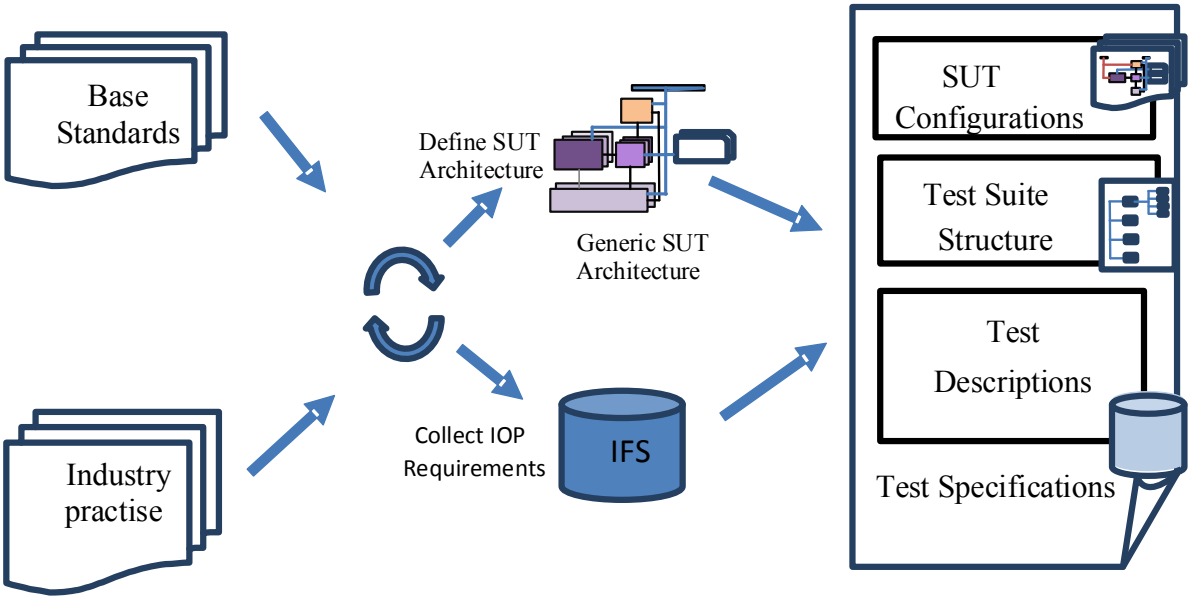


Figure 7.3.1-1: Interoperability Test Specification Development process

7.3.2 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 sections, 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.

7.3.3 Test scenarios

In oneM2M, a large number of use cases is identified. In order to perform interoperability tests, EUTs supporting the same use cases are required. This classification of interoperability tests is given by test scenarios. A test scenario thus selects a set of use cases and is restricted to a sub-set of the full functionality of such a set.

In other words, EUTs considered for defining the test scenarios are implementations of oneM2M entities with various roles, but sharing a common functionality.

In order to cover the test scenarios, different test configurations are defined.

7.3.4 Test bed architecture and Interfaces

A test bed 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 (oneM2M 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.

- **oneM2M architecture element:** It provides oneM2M 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 analyze 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 7.3.4-1 illustrates interfaces involved in the test bed.

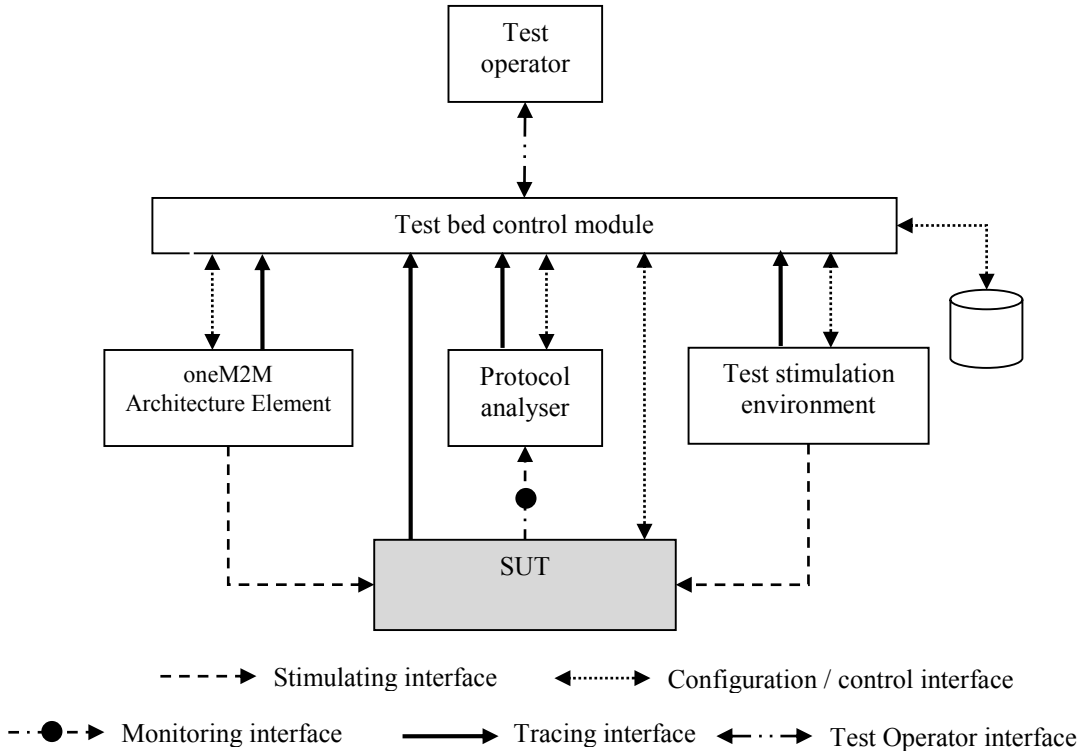


Figure 7.3.4-1: Interfaces of a test bed architecture

7.3.5 Interoperable Functions Statement (IFS)

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

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

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

The guidance to produce IFS proforma is provided in ETSI EG 202 237 [i.3] and no extra guidance is required for the context of oneM2M.

7.3.6 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, etc.) 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. The test descriptions use test configurations in order to cover the different 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 7.3.6-1.

Table 7.3.6-1: Interoperability test description

Identifier	a unique test description ID
Objective	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.

Table 7.3.6-2: Example of Test Description

Interoperability Test Description			
Identifier:			TD_M2M_NH_06
Objective:			AE registers to its registrar CSE via an AE Create Request
Configuration:			M2M_CFG_01
References:			oneM2M TS-0001 [1], clause 10.2.1.1 oneM2M TS-0004 [2], clause 7.3.5.2.1
Pre-test conditions:			• CSEBase resource has been created in CSE with name {CSEBaseName} • AE does not have an AE-ID, i.e. it registers from scratch
Test Sequence			
Step	**RP**	**Type**	**Description**
1		Stimulus	AE is requested to send a AE Create request to register to the Registrar CSE
2	Mca	PRO Check Primitive	• op = 1 (Create) • to = {CSEBaseName} • fr = AE-ID • rqi = (token-string) • ty = 2 (AE) • pc = Serialized representation of <AE> resource
		PRO Check HTTP	Sent request contains • Request method = POST • Request-Target:{CSEBaseName} • Host: IP address or the FQDN of Registrar CSE • X-M2M-RI: (token-string) • X-M2M-Origin: AE-ID • Content-Type: application/vnd.onem2m-res+xml; ty=2 or application/vnd.onem2m-res+json; ty=2 • Message-body: Serialized representation of <AE> resource

Interoperability Test Description			
		PRO Check CoAP	Sent request contains • Method: 0.02 (POST) • Uri-Host: IP address or the FQDN of Registrar CSE • Uri-Path: {CSEBaseName} • Content-type: application/vnd.onem2m-res+xml or application/vnd.onem2m-res+json • oneM2M-TY: 2 • oneM2M-FR: AE-ID • oneM2M-RQI: (token-string) • Payload: Serialized representation of <AE> resource
		PRO Check MQTT	Sent MQTT PUBLISH message: Topic: "/oneM2M/req/<AE-ID>/<Registrar CSE-ID>" Payload: • op = 1 (Create) • to = {CSEBaseName} • fr = AE-ID • rqi = (token-string) • ty = 2 (AE) • pc = Serialized representation of <AE> resource
3		IOP Check	Check if possible that the <AE> resource is created in registrar CSE.
4	Mca	PRO Check Primitive	• rsc = 2001 (CREATED) • rqi = (token-string) same as received in request message • pc = Serialized representation of <AE> resource
		PRO Check HTTP	Registrar CSE sends response containing: • Status Code = 201 (OK) • X-M2M-RSC: 2001 • X-M2M-RI: (token-string) same as received in request message • Content-Location: URI of the created AE resource. • Content-Type: application/vnd.onem2m-res+xml or application/vnd.onem2m-res+json • Message-body: Serialized representation of <AE> resource
		PRO Check CoAP	Registrar sends response containing: • Response Code = 2.01 • oneM2M-RSC: 2001 • oneM2M-RQI: (token-string) same as received in request message • Location-Path: URI of the created AE resource • Payload: Serialized representation of <AE> resource
		PRO Check MQTT	Sent MQTT PUBLISH message: Topic: "/oneM2M/resp/<AE-ID>/<Registrar CSE-ID>" Payload: • to = AE-ID • fr = Registrar CSE-ID • rsc = 2001 (CREATED) • rqi = (token-string) same as received in request message • pc = Serialized representation of <AE> resource
5		IOP Check	AE indicates successful operation

Types of events:

- A **stimulus** corresponds to an event that enforces an DUT to proceed with a specific protocol action, like sending a message for instance.
- A **configure** corresponds to an action to modify the DUT configuration.
- An **IOP check** consists of observing that one DUT behaves as described in the standard: i.e. resource creation, update, deletion, etc… For each IOP check in the Test Sequence, a result can be recorded. The overall **IOP Verdict** will be considered OK if all the IOP checks in the sequence are OK.
- In the context of Interoperability Testing with Conformance Checks, an additional step type, **PRO checks** can be used to verify the appropriate sequence and contents of protocol messages, helpful for debugging purpose. **PRO Verdict** will be PASS if all the PRO checks are PASS.

Annex A (informative): Example of ICS table

A.1 Capability Statement

A list of capabilities defined in the oneM2M TS-0001 [1] are presented in table A.1-1. The capability list can be used to check whether the IUT supports part or whole of the capabilities listed as below.

Table A.1-1: Capabilities for oneM2M Conformance Testing

Item	Capability	Mnemonic	Reference	Status	Support
1	Registration		[1] 10.2.1	C.1	O Yes O No
2	Data Management		[1] 10.2.4, [1] 10.2.19	C.1	O Yes O No
3	Subscription and Notification		[1] 10.2.11	C.2	O Yes O No
4	Group Management		[1] 10.2.7	C.2	O Yes O No
5	Discovery		[1] 10.2.6	C.2	O Yes O No
6	Location Management		[1] 10.2.10	C.2	O Yes O No
7	Device Management		[1]10.2.8	C.2	O Yes O No
8	Communication Management and Delivery Handling		[1] 10.2.5, [1] 10.2.20	C.2	O Yes O No
C.1: Mandatory IF the IUT is declaimed to be developed conforming to oneM2M TS-0001 [1].					
C.2: Optional IF the IUT is declaimed to be developed conforming to oneM2M TS-0001 [1].					

History

Publication history		
V2.0.0	30-Aug-2016	Release 2 - Publication