ETSI CTI Plugtests Guide Draft V0.0.13 (2012-11)

IoT CoAP Plugtests;
Sophia-Antipolis, France;
28 - 30 November 2012

Contents

1 Scope

This document forms the guidelines to lead the technical organization of the 2nd IoT CoAP Plugtests event, in Sophia-Antipolis, from 28th to 30th November 2012. This document is intended to be upgraded for future interoperability events.

This document describes:

- The testbed architecture showing which IoT CoAP systems and components are involved and how they are going to interwork
- The configurations used during test sessions, including the relevant parameter values of the different layers
- The interoperability test descriptions, describing the scenarios, which the participants will follow to perform the interoperability tests

2 References

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

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

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

2.1 Normative references

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

[1] Constrained Application Protocol (CoAP); draft-ietf-core-coap-12

[2] Core Link Format; RFC 6690

[3] Observing Resources in CoAP; draft-ietf-core-observe-07

[4] Blockwise transfers in CoAP; draft-ietf-core-block-10

[5] ETSI TS 102 921: "Machine-to-Machine Communications (M2M); mIa, dIa and mId interfaces".

[6] ETSI TS 102 690: "Machine to Machine Communications (M2M); Functional Architecture".

3 Abbreviations

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

ACK	Acknowledgement
aPoC	The Application Point of Contract is a URI that identifies how requests are re-targeted
CON	Confirmable
DA	Device Application
dIa	device application Interface

Device' (D')	Hosts DA that communicates to a GSCL using the dIa reference point.
GSCL	Gateway SCL
mIa	M2M application Interface
mId	M2M device Interface
NON	Non-Confirmable
NA	Network Application
NSCL	Network SCL
RST	Reset
SCL	Service Capability Layer
TD	Test Description

4 Conventions

4.1 Interoperability test process

4.1.1 Introduction

The goal of interoperability test is to check that devices resulting from protocol implementations are able to work together and provide the functionalities provided by the protocols. As necessary, a message may be checked during an interoperability test, when a successful functional verification may result from an incorrect behaviour for instance. Detailed protocol checks are part of the conformance testing process and are thus avoided during the Interoperability tests.

The test session will be mainly executed between 2 devices from different vendors. For some test purposes, it may be necessary to have more than 2 devices involved. The information about the test configuration like the number of devices or the roles required are indicated in the test description tables below.

4.1.2 The test description proforma

The test descriptions are provided in proforma tables. The following different types of test operator actions are considered during the test execution:

- A **stimulus** corresponds to an event that enforces an EUT to proceed with a specific protocol action, like sending a message for instance
- A **verify** consists of verifying that the EUT behaves according to the expected behaviour (for instance the EUT behaviour shows that it receives the expected message)
- A **configure** corresponds to an action to modify the EUT configuration
- A **check** ensures the correctness of protocol messages on reference points, with valid content according to the specific interoperability test purpose to be verified.

For the execution of the interoperability test sessions, the following conventions apply:

- Every ‘Check’ step of a test description should be performed using a trace created by a monitor tool (see clause ‘Tooling’ below) and may be skipped due to time restrictions

4.2 Tooling

- Participant shall use their own tools (e.g. tcpdump, wireshark) for logging and analysing messages for the “check” purposes

- Participants will be given the opportunity to upload their log files to a central server for a format validity check. The checks defined in each test description will be automatically performed by the central server

- Except for the "check" events, the verification of the message correctness is not part of the Interoperability test process

- To realize the lossy context of tests TD_XXX (e.g. packet loss and packet delay) a gateway will be provided which will serve as an intermediate between the client and the server to simulate the lossy medium (technically this is implemented using NAT-style UDP port redirections)

4.3 Test Description naming convention

Table 1: TD naming convention

TD/<root>/<gr>/<nn>		
<root> = root	COAP	Constrained Application Protocol
	M2M_COAP	CoAP Binding for M2M
<gr> = group	CORE	Core protocol
	LINK	Core Link Format
	BLOCK	Blockwise transfers
	OBS	Observing Resources
<nn> = sequential number		01 to 99

4.4 Test Summary – Mandatory CoAP Tests

Table 2: Mandatory Tests

1	TD_COAP_CORE_01	Perform GET transaction (CON mode)
2	TD_COAP_CORE_02	Perform DELETE transaction (CON mode)
3	TD_COAP_CORE_03	Perform PUT transaction (CON mode)
4	TD_COAP_CORE_04	Perform POST transaction (CON mode)
5	TD_COAP_CORE_05	Perform GET transaction (NON mode)
6	TD_COAP_CORE_06	Perform DELETE transaction (NON mode)
7	TD_COAP_CORE_07	Perform PUT transaction (NON mode)
8	TD_COAP_CORE_08	Perform POST transaction (NON mode)
9	TD_COAP_CORE_09	Perform GET transaction with separateresponse (CON mode, no piggyback)
10	TD_COAP_CORE_10	Perform GET transaction containing Token option (CON mode)
11	TD_COAP_CORE_11	Perform GET transaction containing token option with a separate response (CON mode)
12	TD_COAP_CORE_12	Perform GET transaction not containing Token option (CON mode)
13	TD_COAP_CORE_13	Perform GET transaction containing several URI-Path options (CON mode)
14	TD_COAP_CORE_14	Perform GET transaction containing several URI-Query options (CON mode)
15	TD_COAP_CORE_15	Perform GET transaction (CON mode, piggybacked response) in a lossy context
16	TD_COAP_CORE_16	Perform GET transaction (CON mode, delayed response) in a lossy context
17	TD_COAP_CORE_17	Perform GET transaction with a separate response (NON mode)
18	TD_COAP_CORE_18	Perform POST transaction with responses containing several Location-Path options (CON mode)
19	TD_COAP_CORE_19	Perform POST transaction with responses containing several Location-Query options (CON mode)
20	TD_COAP_CORE_20	Perform GET transaction containing the Accept option (CON mode)
21	TD_COAP_CORE_21	Perform GET transaction containing the ETag option (CON mode)
22	TD_COAP_CORE_22	Perform GET transaction with responses containing the ETag option and requests containing the If-Match option (CON mode)
23	TD_COAP_CORE_23	Perform PUT transaction containing the If-None-Match option (CON mode)

4.5 Test Summary – Optional CoAP Tests

Table 3: Optional Tests

1	TD_COAP_LINK_01	Access to well-known interface for resource discovery
2	TD_COAP_LINK_02	Use filtered requests for limiting discovery results
3	TD_COAP_LINK_03	Handle empty prefix value strings
4	TD_COAP_LINK_04	Filter discovery results in presence of multiple rt attributes
5	TD_COAP_LINK_05	Filter discovery results using if attribute and prefix value strings
6	TD_COAP_LINK_06	Filter discovery results using sz attribute and prefix value strings
7	TD_COAP_LINK_07	Filter discovery results using href attribute and complete value strings
8	TD_COAP_LINK_08	Filter discovery results using href attribute and prefix value strings
9	TD_COAP_LINK_09	Arrange link descriptions hierarchically
10	TD_COAP_BLOCK_01	Handle GET blockwise transfer for large resource (early negotiation)
11	TD_COAP_BLOCK_02	Handle GET blockwise transfer for large resource (late negotiation)
12	TD_COAP_BLOCK_03	Handle PUT blockwise transfer for large resource
13	TD_COAP_BLOCK_04	Handle POST blockwise transfer for large resource
14	TD_COAP_OBS_01	Handle resource observation with CON messages
15	TD_COAP_OBS_02	Handle resource observation with NON messages
16	TD_COAP_OBS_03	Stop resource observation
17	TD_COAP_OBS_04	Client detection of deregistration (Max-Age)
18	TD_COAP_OBS_05	Server detection of deregistration (client OFF)
19	TD_COAP_OBS_06	Server detection of deregistration (explicit RST)
20	TD_COAP_OBS_07	Server cleans the observers list on DELETE
21	TD_COAP_OBS_08	Server cleans the observers list when observed resource content-format changes
22	TD_COAP_OBS_09	Update of the observed resource
23	TD_COAP_CORE_24	Perform POST transaction with responses containing several Location-Path options (Reverse Proxy in CON mode)
24	TD_COAP_CORE_25	Perform POST transaction with responses containing several Location- Query (Reverse proxy)
25	TD_COAP_CORE_26	Perform GET transaction containing the Accept option (CON mode) (Reverse proxy)
26	TD_COAP_CORE_27	Perform GET transaction with responses containing the ETag option and requests containing the If-Match option (CON mode) (Reverse proxy)
27	TD_COAP_CORE_28	Perform GET transaction with responses containing the ETag option and requests containing the If-None-Match option (CON mode) (Reverse proxy)
28	TD_COAP_CORE_29	Perform GET transaction with responses containing the Max-Age option (Reverse proxy)

4.6 CoAP Binding for M2M REST Resources

Table 4: CoAP Binding for M2M REST Resources

1	TD_M2M_COAP_01	M2M DA registers to its local SCL via an applicationCreateRequest (CoAP POST)
2	TD_M2M_COAP_02	M2M DA retrieves application resource via an applicationRetrieveRequest (CoAP GET)
3	TD_M2M_COAP_03	M2M DA updates attribute in application resource via an applicationUpdateRequest (CoAP PUT)
4	TD_M2M_COAP_04	M2M DA creates a subscription to application resource via subscriptionCreateRequest (CoAP POST)
5	TD_M2M_COAP_05	M2M GSCL sends notification(s) via subscriptionNotifyRequest (CoAP POST)
6	TD_M2M_COAP_06	M2M DA cancels subscription via an subscriptionDeleteRequest (CoAP DELETE)
7	TD_M2M_COAP_07	M2M DA de-registers by deleting application resource via an applicationDeleteRequest (CoAP DELETE)
8	TD_M2M_COAP_08	Handle contentInstanceRetrieveRequest with targetID containing several path segments
9	TD_M2M_COAP_09	Handle contentInstanceRetrieveRequest with targetID containing several query options
10	TD_M2M_COAP_10	Handle contentInstanceRetrieveRequest with targetID using partial addressing
11	TD_M2M_COAP_11	M2M DA registration with Announcement
12	TD_M2M_COAP_12	M2M NA multi-hop resource retrieval using Proxy-URI (CoAP proxy)
13	TD_M2M_COAP_13	M2M NA multi-hop resource retrieval using m2mPocs (M2M proxy)

5 Basic Configuration

5.1 Resources offered by servers under test

In order to ease test setup and execution, CoAP servers are requested to support the following resources and primitives:

Table 5: M2M Primitives

Subject	Primitive
Application management	applicationCreateRequest / Response
	applicationRetrieveRequest / Response
	applicationUpdateRequest / Response
	applicationDeleteRequest / Response
Subscription management	subscriptionCreateRequest / Response
	subscriptionNotifyRequest / Response
	subscriptionDeleteRequest / Response
Content management	containerCreateRequest/Response
	contentInstanceCreateRequest / Response

	contentInstanceRetrieveRequest / Response
Announcement management	applicationAnncCreateRequest / Response
PoC management	m2mPocCreateRequest / Response

Table 64: M2M Primitive Resource Representations

M2M Primitive	Resource name	Resource Representation
applicationCreateRequest	<app>	<?xml version="1.0"?> <tns:application xmlns:tns=”http://uri.etsi.org/m2m” tns:id=”app”/>
applicationCreateResponse	<app>	<?xml version="1.0"?> <tns:application xmlns:tns="http://uri.etsi.org/m2m" tns:id="app "> <tns:expirationTime>*2012-10-25T13:13:04*</tns:expirationTime> </tns:application>
applicationRetrieveResponse	<app>	<?xml version="1.0"?> <tns:application xmlns:tns="http://uri.etsi.org/m2m" tns:id="app"> <tns:applicationStatus>*ONLINE*</tns:applicationStatus> <tns:expirationTime>*2012-11-19T18:39:05*</tns:expirationTime> <tns:lastModifiedTime>*2012-11-12T19:59:05*</tns:lastModifiedTime> <tns:containersReference> */gsclBase/applications/app/containers* </tns:containersReference> <tns:groupsReference> */gsclBase/applications/app/groups* </tns:groupsReference> <tns:accessRightsReference> */gsclBase/applications/app/accessRights* </tns:accessRightsReference> <tns:subscriptionsReference>/ */gsclBase/applications/app/subscriptions* </tns:subscriptionsReference> </tns:application>
applicationUpdateRequest	<app>	<?xml version="1.0"?> <tns:application xmlns:tns="http://uri.etsi.org/m2m"> <tns:aPoc>*coap://DA_IP_Address:Port*</tns:aPoc> </tns:application>
applicationUpdateResponse	<app>	<?xml version="1.0"?> <tns:application xmlns:tns=”http://uri.etsi.org/m2m”> <tns:expirationTime>*2012-10-25T13:13:04*</tns:expirationTime> </tns:application>
applicationCreateRequest	<app_ann>	<?xml version="1.0"?> <tns:application xmlns:tns=”http://uri.etsi.org/m2m” tns:id="app_ann"> <tns:announceTo> <tns:activated>true</tns:activated> </tns:announceTo> <tns:aPoc>coap://*DA_IP_Address:Port*</tns:aPoc> </tns:application>
applicationCreateResponse	<app_ann>	<tns:application xmlns:tns="http://uri.etsi.org/m2m" tns:id="app_ann "> <tns:expirationTime>*2012-10-25T13:13:04*</tns:expirationTime> </tns:application>

subscriptionCreateRequest	<sub>	<?xml version="1.0"?> <tns:subscription xmlns:tns="http://uri.etsi.org/m2m" tns:id=”sub”> <tns:contact>coap://*DA_IP_Addr:Port/da_notif*</tns:contact> </tns:subscription>
subscriptionCreateResponse	<sub>	<?xml version="1.0"?> <tns:subscription xmlns:tns="http://uri.etsi.org/m2m" tns:id="sub "> <tns:expirationTime>*2012-10-25T13:13:04*</tns:expirationTime> </tns:subscription>
subscriptionNotifyRequest	<sub>	<?xml version="1.0"?> <tns:notify xmlns:tns="http://uri.etsi.org/m2m"> <statusCode>1</statusCode> <representation> *base64Binary encoded representation of application resource* </representation> <subscriptionReference> *coap://GW_IP_Addr:Port/gw01/applications/app/subscriptions/sub* </subscriptionReference> </tns:notify>
subscriptionNotifyResponse	<sub>	<?xml version="1.0"?> <tns:notify xmlns:tns="http://uri.etsi.org/m2m"> <statusCode>1</statusCode> </tns:notify>
containerCreateRequest	<container1>	<?xml version="1.0"?> <tns:container xmlns:tns=”http://uri.etsi.org/m2m” tns:id=”container1”/>
containerCreateResponse	<container1>	<?xml version="1.0"?> <tns:container xmlns:tns=”http://uri.etsi.org/m2m” tns:id=”container1”/>
contentInstanceCreateReques t	<test>	<?xml version="1.0"?> <tns:contentInstance xmlns:tns="http://uri.etsi.org/m2m"> <tns:content> <tns:textContent>*content*</tns:textContent> </tns:content> </tns:contentInstance>
contentInstanceCreateRespon se	<test>	<?xml version="1.0"?> <tns:contentInstance xmlns:tns=”http://uri.etsi.org/m2m” tns:id="test"/>

Table 7: Resources offered by CoAP Servers

Resource name	Description	Used in
/test	Default test resource	TD_COAP_CORE_01 TD_COAP_CORE_02 TD_COAP_CORE_03 TD_COAP_CORE_04 TD_COAP_CORE_05 TD_COAP_CORE_06 TD_COAP_CORE_07 TD_COAP_CORE_08 TD_COAP_CORE_10 TD_COAP_CORE_11 TD_COAP_CORE_14 TD_COAP_CORE_18 TD_COAP_CORE_22 TD_COAP_LINK_08 TD_COAP_LINK_10
/validate	Resource which varies	TD_COAP_CORE_21 TD_COAP_CORE_27

		TD_COAP_CORE_29
/create1	Resource which doesn't exist yet (to perform atomic PUT)	TD_COAP_CORE_23
/create2	Resource which doesn't exist yet	TD_COAP_CORE_24
/create3	Resource which doesn't exist yet	TD_COAP_CORE_28
/seg1/seg2/seg3	Long path resource	TD_COAP_CORE_12
/location1/location2/location3	Location path resource	TD_COAP_CORE_18 TD_COAP_CORE_24
/location-query	Resource accepting location query parameters	TD_COAP_CORE_19 TD_COAP_CORE_25
/query	Resource accepting query parameters	TD_COAP_CORE_13
/separate	Resource which cannot be served immediately and which cannot be acknowledged in a piggy-backed way	TD_COAP_CORE_09 TD_COAP_CORE_15 TD_COAP_CORE_16
/large	Large resource	TD_COAP_BLOCK_01 TD_COAP_BLOCK_02
/large-update	Large resource that can be updated using PUT method	TD_COAP_BLOCK_03
/large-create	Large resource that can be created using POST method	TD_COAP_BLOCK_04
/obs	Observable resource which changes every 5 seconds and for which the server is configured to send confirmable (CON) notifications	TD_COAP_OBS_01 TD_COAP_OBS_03 TD_COAP_OBS_04 TD_COAP_OBS_05 TD_COAP_OBS_06 TD_COAP_OBS_07 TD_COAP_OBS_08 TD_COAP_OBS_09
/obs-non	Observable resource which changes every 5 seconds and for which the server is configured to send non-confirmable (NON) notifications	TD_COAP_OBS_02
/.well-known/core	Core Link Format	TD_COAP_LINK_01 TD_COAP_LINK_02 TD_COAP_LINK_03 TD_COAP_LINK_04 TD_COAP_LINK_05 TD_COAP_LINK_06 TD_COAP_LINK_07 TD_COAP_LINK_08 TD_COAP_LINK_09 TD_COAP_LINK_10
/multi-format	Resource that exists in different content formats (text/plain utf8 and application/xml)	TD_COAP_CORE_20 TD_COAP_CORE_26
/link1	Link test resource	TD_COAP_LINK_07 TD_COAP_LINK_08
/link2	Link test resource	TD_COAP_LINK_07 TD_COAP_LINK_08

/link3	Link test resource	TD_COAP_LINK_07 TD_COAP_LINK_08
/path	Hierarchical link description entry	TD_COAP_LINK_09
/path/sub1	Hierarchical link description sub-resource	TD_COAP_LINK_09
/path/sub2	Hierarchical link description sub-resource	TD_COAP_LINK_09
/path/sub3	Hierarchical link description sub-resource	TD_COAP_LINK_09
/alternate	Alternate	TD_COAP_LINK_10

Note on resource sizes:

- Resources used in TD_COAP_CORE tests should not exceed 64 bytes
- Large resources used in TD_COAP_BLOCK tests shall not exceed 2048 bytes
- TD_COAP_LINK tests may require usage of Block options with some implementations

5.2 M2M Access Control

M2M Access control is not being used. Hence there is no primitive attribute ‘requestingEntity’ being mapped to any CoAP query parameter.

5.3 aPoc Re-Targeting Procedure

When M2M DA registers to its GSCL it can

- either use the aPoc Re-Targeting mechanism
- or create and update contentInstance resource on the GSCL

As a consequence, when the GSCL receives a resource retrieve request, it will

- either forward the request to DA
- or reply directly to the request itself

5.4 CoAP settings

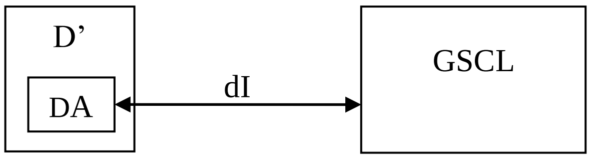
Unless stated otherwise, the following settings shall be applied:

- Each equipment under test shall be configured with a unicast address
- Client cache shall be cleaned up after each test
- Use of ETag option shall be avoided, but implementation should be prepared to handle it
- Use of Token shall be avoided, but implementation should be prepared to handle it
- Use of Piggybacked responses shall be preferred

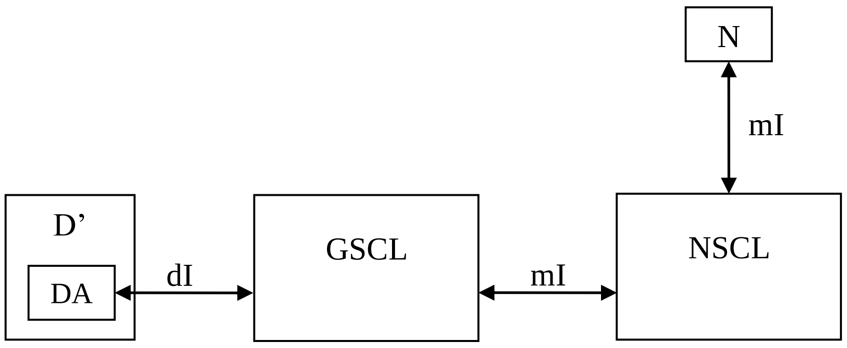
6 Test Configurations

This section defines the different test configurations.

6.1 Basic M2M CoAP (M2M_CFG_01)



6.2 M2M CoAP Multihop (M2M_CFG_02)



6.3 Basic CoAP 1 (CoAP_CFG_01)

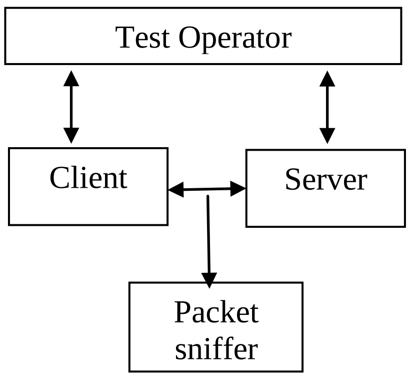


Figure 1: Basic One-2-One CoAP client/server Configuration

6.4 CoAP in lossy context (CoAP_CFG_02)

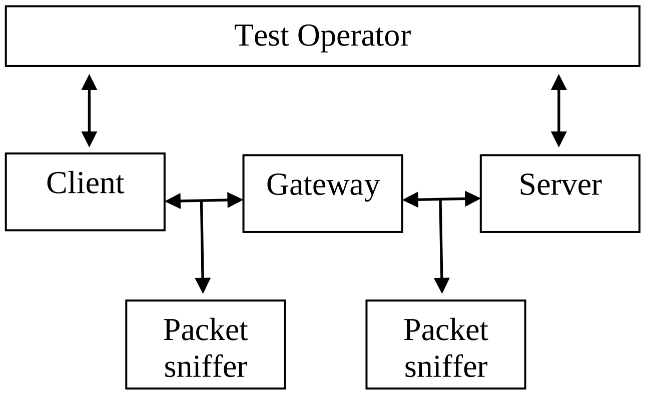


Figure 2: Basic One-2-One CoAP client/server Configuration in lossy context

The Gateway emulates a lossy medium between the client and the server. It does not implement the CoAP protocol itself (in other terms it is not a CoAP proxy), but works at the transport layer. It provides two features:

- It performs NAT-style UDP port redirections towards the server (thus the client contacts the gateway and is transparently redirected towards the server)
- It randomly drops packets that are forwarded between the client and the server

6.5 Test Configuration 3 (CoAP_CFG_03)

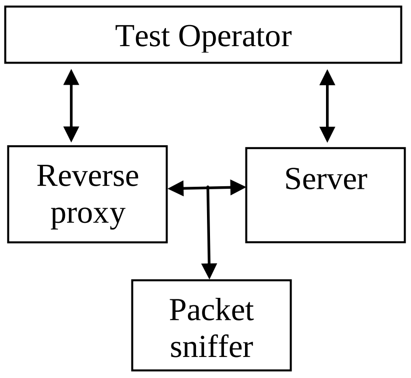


Figure 3: Basic One-2-One CoAP proxy/server Configuration

The reverse proxy shown in the Figure 3 is assumed as CoAP/CoAP proxy. Test operator includes an interface (it can be a CoAP client) that creates the stimulus to initiate the tests for reverse proxy.

More clearly, there exists two methods to create the stimulus for reverse proxy.

1. Reverse proxy can provide a direct interface to create and launch the stimulus
2. A CoAP client can be connected to reverse proxy to create and launch the stimulus for the tests

In the both cases, reverse proxy and client equally act as point of observation.

7 CoAP Scenarios

This section describes the different test scenarios. To ensure the good execution of these scenarios, it is assumed that the following settings are applied before each test execution:

- Each equipment under test shall be configured with a unicast address
- Client cache shall be cleaned up
- Use of ETag option shall be avoided except if explicitly stated in the test description, but implementation should be prepared to handle it
- Use of Token option shall be avoided except if explicitly stated in the test description, but implementation should be prepared to handle it
- Use of Piggybacked responses shall be preferred unless stated otherwise in the test description

7.1 CoAP protocol

Interoperability Test Description			
Identifier:	TD_COAP_CORE_01		
Objective:	Perform GET transaction (CON mode)		
Configuration:	CoAP_CFG_01		
References:	[1] 5.8.1,1.2,2.1,2.2,3.1		
Pre-test conditions:	• Server offers the resource **/test** with resource content is not empty that handles GET with an arbitrary payload		
Test Sequence:	**Step**	**Type**	**Description**
	1	Stimulus	Client is requested to send a GET request with: • Type = 0(CON) • Code = 1(GET)
	2	Check	The request sent by the client contains: • Type=0 and Code=1
	3	Check	Server sends response containing: • Code = 69(2.05 Content) • The same Message ID as that of the request sent by the client • Content format option
	4	Verify	Client displays the received information

Interoperability Test Description			
Identifier:	TD_COAP_CORE_02		
Objective:	Perform DELETE transaction (CON mode)		
Configuration:	CoAP_CFG_01		
References:	[1] 5.8.4,1.2,2.1,2.2,3.1		
Pre-test conditions:	• Server offers a **/test** resource that handles DELETE		
Test Sequence:	**Step**	**Type**	**Description**
	1	Stimulus	Client is requested to send a DELETE request with: • Type = 0(CON) • Code = 4(DELETE)
	2	Check	The request sent by the client contains: • Type=0 and Code=4
	3	Check	Server sends response containing: • Code = 66(2.02 Deleted) • The same Message ID as that of the request sent by the client
	4	Verify	Client displays the received information

Interoperability Test Description			
Identifier:	TD_COAP_CORE_03		
Objective:	Perform PUT transaction (CON mode)		
Configuration:	CoAP_CFG_01		
References:	[1] 5.8.3,1.2,2.1,2.2,3.1		
Pre-test conditions:	• Server offers already available resource **/test or accepts creation of new resource on /test** that handles PUT		
Test Sequence:	**Step**	**Type**	**Description**
	1	Stimulus	Client is requested to send a PUT request with: • Type = 0(CON) • Code = 3(PUT) • An arbitrary payload • Content format option
	2	Check	The request sent by the client contains: • Type=0 and Code=3
	3	Verify	Server displays received information
	4	Check	Server sends response containing: • Code = 68 (2.04 Changed) or 65 (2.01 Created) • The same Message ID as that of the request by the client
	5	Verify	Client displays the received response

Interoperability Test Description			
Identifier:	TD_COAP_CORE_04		
Objective:	Perform POST transaction (CON mode)		
Configuration:	CoAP_CFG_01		
References:	[1] 5.8.2,1.2,2.1,2.2,3.1		
Pre-test conditions:	• Server accepts creation of new resource on / **test** (resource does not exist yet)		
Test Sequence:	**Step**	**Type**	**Description**
	1	Stimulus	Client is requested to send a POST request with: • Type = 0(CON) • Code = 2(POST) • An arbitrary payload • Content format option
	2	Check	The request sent by the client contains: • Type=0 and Code=2
	3	Verify	Server displays received information
	4	Check	Server sends response containing: • Code = 65(2.01 Created) or 68 (2.04 changed) • The same Message ID as that of the request sent by the client
	5	Verify	Client displays the received response

Interoperability Test Description			
Identifier:	TD_COAP_CORE_05		
Objective:	Perform GET transaction (NON mode)		
Configuration:	CoAP_CFG_01		
References:	[1] 5.8.1, 5.2.3		
Pre-test conditions:	• Server offers a **/test** resource with resource content is not empty that handles GET		
Test Sequence:	**Step**	**Type**	**Description**
	1	Stimulus	Client is requested to send a GET request with: • Type = 1(NON) • Code = 1(GET)
	2	Check	The request sent by the client contains: • Type=1 and Code=1
	3	Check	Server sends response containing: • Type = 1(NON) • Code= 69(2.05 Content) • Content format option
	4	Verify	Client displays the received information

Interoperability Test Description			
Identifier:	TD_COAP_CORE_06		
Objective:	Perform DELETE transaction (NON mode)		
Configuration:	CoAP_CFG_01		
References:	[1] 5.8.4,5.2.3		
Pre-test conditions:	• Server offers a **/test** resource that handles DELETE		
Test Sequence:	**Step**	**Type**	**Description**
	1	Stimulus	Client is requested to send a DELETE request with: • Type = 1(NON)

Interoperability Test Description			
			• Code = 4(DELETE)
	2	Check	The request sent by the client contains: • Type=1 and Code=4
	3	Check	Server sends response containing: • Type = 1(NON) • Code = 66(2.02 Deleted)
	4	Verify	Client displays the received information

Interoperability Test Description			
Identifier:	TD_COAP_CORE_07		
Objective:	Perform PUT transaction (NON mode)		
Configuration:	CoAP_CFG_01		
References:	[1] , 5.8.3, 5.2.3		
Pre-test conditions:	• Server offers a ***/test*** resource that handles PUT		
Test Sequence:	**Step**	**Type**	**Description**
	1	Stimulus	Client is requested to send a PUT request with: • Type = 1(NON) • Code = 3(PUT) • An arbitrary payload • Content format option
	2	Check	The request sent by the client contains: • Type=1 and Code=3
	3	Verify	Server displays the received information
	4	Check	Server sends response containing: • Type = 1(NON) • Code = 68 (2.04 Changed) or 65 (2.01 Created)
	5	Verify	Client displays the received response

Interoperability Test Description			
Identifier:	TD_COAP_CORE_08		
Objective:	Perform POST transaction (NON mode)		
Configuration:	CoAP_CFG_01		
References:	[1] 5.8.2,5.2.3		
Pre-test conditions:	• Server accepts creation of new resource on **/test** (resource does not exist yet)		
Test Sequence:	**Step**	**Type**	**Description**
	1	Stimulus	Client is requested to send a POST request with: • Type = 1(NON) • Code = 2(POST) • An arbitrary payload • Content format option
	2	Check	The request sent by the client contains: • Type=1 and Code=2
	3	Verify	Server displays the received information
	4	Check	Server sends response containing: • Type = 1(NON) • Code = 65(2.01 Created) or 68 (2.04 changed)
	5	Verify	Client displays the received response

Interoperability Test Description			
Identifier:	TD_COAP_CORE_09		
Objective:	Perform GET transaction with separate response (CON mode, no piggyback)		
Configuration:	CoAP_CFG_01		
References:	[1] clause 5.8.1,5.2.2		
Pre-test conditions:	• Server offers a resource **/separate** which cannot be served immediately and which cannot be acknowledged in a piggybacked way.		
Test Sequence:	**Step**	**Type**	**Description**
	1	Stimulus	Client is requested to send a confirmable GET request to server's resource
	2	Check	The request sent by the client contains: • Type = 0 (CON) • Code = 1 (GET) • Client generated Message ID
	3	Check	Server sends response containing: • Type = 2 (ACK) • Code = 0 • Same message ID as in the request sent by the client • empty Payload
	4	Check	Server sends response containing: • Type = 0 (CON) • Code = 69 (2.05 content) • Server generated Message ID • Not empty Payload Content format option
	5	Check	Client sends response containing: • Type = 2 (ACK) • Code = 0 • Same message ID as in the response sent by the server in step 4 • empty Payload
	6	Verify	Client displays the response

Interoperability Test Description
Note: Steps 3 and 4 may occur out-of-order

Interoperability Test Description			
Identifier:	TD_COAP_CORE_10		
Objective:	Perform GET transaction containing Token option (CON mode)		
Configuration:	CoAP_CFG_01		
References:	[1] clause 2.2 ,5.8.1, 5.10.1		
Pre-test conditions:	• Server offers a **/test** resource with resource content is not empty that handles GET		
Test Sequence:	**Step**	**Type**	**Description**
	1	Stimulus	Client is requested to send a GET request to server's resource including Token option
	2	Check	The request sent by the client contains: • Type = 0 (CON) • Code = 1 (GET) • Option Type = Token • Token value = a value generated by the client • Length of the token should be between 1 to 8 B
	3	Check	Server sends response containing: • Code = 69 (2.05 content) • Length of the token should be between 1 to 8 B • Token = the same value as in the request sent by the client • Not empty Payload • Content format option
	4	Verify	Client displays the response

Interoperability Test Description			
Identifier:	TD_COAP_CORE_11		
Objective:	Perform GET transaction containing token option with a separate response (CON mode)		
Configuration:	CoAP_CFG_01		
References:	[1] clause 2.2, 5.2.2, 5.8.1		
Pre-test conditions:	• Server offers a resource **/separate** which cannot be served immediately.		
Test Sequence:	**Step**	**Type**	**Description**
	1	Stimulus	Client is requested to send a GET request to server's resource including Token option
	2	Check	The request sent by the client contains: • Type = 0 (CON) • Code = 1 (GET) • Option Type = Token • Token value = a value generated by the client • Length of the token should be between 1 to 8 B •
	3	Check	Server sends acknowledgement containing: • Type = 2 (ACK) • Code = 0 (Empty) • same Message-Id as in step 2 • empty Payload
	4	Check	Server sends response containing: • Type = 0 (CON) • Code = 69 (2.05 content) • Length of the token should be between 1 to 8 B

Interoperability Test Description			
			• Token value = the same value as in the request sent by the client in step 2 • Not empty Payload
	5	Check	Client sends acknowledgement containing: • Type = 2 (ACK) • Code = 0 (Empty) • same Message-Id as in step 4 • empty Payload
	6	Verify	Client displays the response

Interoperability Test Description			
Identifier:	TD_COAP_CORE_12		
Objective:	Perform GET transaction not containing Token option (CON mode)		
Configuration:	CoAP_CFG_01		
References:	[1] clause 2.2 ,5.8.1, 5.10.1		
Pre-test conditions:	• Server offers a ***/test*** resource with resource content is not empty that handles GET		
Test Sequence:	**Step**	**Type**	**Description**
	1	Stimulus	Client is requested to send a confirmable GET request not containing Token option to server's resource
	2	Check	The request sent by the client contains: • Type = 0 (CON) • Code = 1 (GET) • No Token option
	3	Check	Server sends response containing: • Code = 69 (2.05 content) • No Token option • Not empty Payload • Content format option
	4	Verify	Client displays the response

Interoperability Test Description			
Identifier:	TD_COAP_CORE_13		
Objective:	Perform GET transaction containing several URI-Path options (CON mode)		
Configuration:	CoAP_CFG_01		
References:	[1] clause 5.4.5, 5.10.2,6.5		
Pre-test conditions:	• Server offers a ***/seg1/seg2/seg3*** resource with resource content is not empty		
Test Sequence:	**Step**	**Type**	**Description**
	1	Stimulus	Client is requested to send a confirmable GET request to server's resource
	2	Check	The request sent by the client contains: • Type = 0 (CON) • Code = 1 (GET) • Option type = URI-Path (one for each path segment), not containing **'/'** symbol
	3	Check	Server sends response containing: • Code = 69 (2.05 content) • Not empty Payload • Content format option
	4	Verify	Client displays the response

Interoperability Test Description			
Identifier:	TD_COAP_CORE_14		
Objective:	Perform GET transaction containing several URI-Query options (CON mode)		
Configuration:	CoAP_CFG_01		
References:	[1] clause 5.4.5, 5.10.2,6.5		
Pre-test conditions:	• Server offers a ***/query*** resource with resource content is not empty		
Test Sequence:	**Step**	**Type**	**Description**
	1	Stimulus	Client is requested to send a confirmable GET request with three Query parameters (e.g. ?first=1&second=2&third=3) to the server's resource
	2	Check	The request sent by the client contains: • Type = 0 (CON) • Code = 1 (GET) • Option type = URI-Query (More than one query parameter)
	3	Check	Server sends response containing: • Type = 0 (CON) or 2 (ACK) • Code = 69 (2.05 content) • Not empty Payload Content format option
	4	Verify	Client displays the response

Interoperability Test Description			
Identifier:	TD_COAP_CORE_15		
Objective:	Perform GET transaction (CON mode, piggybacked response) in a lossy context		
Configuration:	CoAP_CFG_02		
References:	[1] clause 4.4.1, 5.2.1,5.8.1		
Pre-test conditions:	• Gateway is introduced and configured to produce packet losses • Server offers a ***/test*** resource with resource content is not empty that can handle GET		
Test Sequence:	**Step**	**Type**	**Description**
	1	Stimulus	Client is requested to send a confirmable GET request to server's resource
	2	Check	Sent request must contain: • Type = 0 • Code = 1 • Client generated Message ID
	3	Check	Server sends response containing: • Type = 2 (ACK) • Code = 69 (2.05 content) • Not empty Payload • Content format option
	4	Verify	Client displays the response
	5	Check	Repeat steps 1-4 until at least one of the following actions has been observed: • One dropped request • One dropped response
	6	Verify	For each case mentioned in step 5: Observe that retransmission is launched

Interoperability Test Description			
Identifier:	TD_COAP_CORE_16		
Objective:	Perform GET transaction (CON mode, delayed response) in a lossy context		
Configuration:	CoAP_CFG_02		
References:	[1] clause 4.4.1, 5.2.2,5.8.1		
Pre-test conditions:	• Gateway is introduced and configured to produce packet losses • Server offers a **/separate** resource which cannot be served immediately and which cannot be acknowledged in a piggybacked way.		
Test Sequence:	**Step**	**Type**	**Description**
	1	Stimulus	Client is requested to send a confirmable GET request to server's resource
	2	Check	The requested sent by the client contains: • Type = 0 • Code = 1 • a message ID generated by the client
	3	Check	Server sends response containing: • Type = 2 (ACK) • message ID is the same as in the request • empty Payload
	4	Check	Server sends response containing: • Type = 0 (CON) • Code = 69 (2.05 content) • Not empty Payload • Content format option
	5	Check	Client sends response containing: • Type = 2 (ACK) • message ID is the same as in the response of step 3 • empty Payload
	6	Verify	Client displays the response
	7	Check	Repeat steps 1-6 until at least one of the following actions has been observed: • One dropped request • One dropped request ACK • One dropped response • One dropped response ACK and its retransmission
	8	Verify	For each case mentioned in step 7: Observe that retransmission is launched

Interoperability Test Description			
Identifier:	TD_COAP_CORE_17		
Objective:	Perform GET transaction with a separate response (NON mode)		
Configuration:	CoAP_CFG_01		
References:	[1] clause 2.2, 5.2.2, 5.8.1		
Pre-test conditions:	• Server offers a resource **/separate** which cannot be served immediately.		
Test Sequence:	**Step**	**Type**	**Description**
	1	Stimulus	Client is requested to send a non-confirmable GET request to server's resource
	2	Check	The request sent by the client contains: • Type = 1 (NON) • Code = 1 (GET) • A message ID generated by the Client
	3	Check	Server DOES NOT send response containing: • Type = 2 (ACK) • Same message ID as in the request in step 2 • empty Payload
	4	Check	Server sends response containing: • Type = 1 (NON) • Code = 69 (2.05 content)

Interoperability Test Description			
			• Not empty Payload d • Content format option
	5	Verify	Client displays the response

Interoperability Test Description			
Identifier:	TD_COAP_CORE_18		
Objective:	Perform POST transaction with responses containing several Location-Path options (CON mode)		
Configuration:	CoAP_CFG_01		
References:	[1] clause 5.8.1,5.10.8,5.9.1.1		
Pre-test conditions:	• Server accepts creation of new resource on **/test**and the created resource is located at **/location1/location2/location3** (resource does not exist yet)		
Test Sequence:	**Step**	**Type**	**Description**
	1	Stimulus	Client is requested to send a confirmable POST request to server's resource
	2	Check	The request sent by the client contains: • Type = 0 (CON • Code = 2 (POST) • An arbitrary payload • Content-format option
	3	Check	Server sends response containing: • Code = 65 (2.01 created) • Option type = Location-Path (one for each segment) • Option values must contain "location1", "location2" & "location3" without containing any '/'
	4	Verify	Client displays the response

Interoperability Test Description			
Identifier:	TD_COAP_CORE_19		
Objective:	Perform POST transaction with responses containing several Location-Query options (CON mode)		
Configuration:	CoAP_CFG_01		
References:	[1] clause 5.8.1,5.10.8,5.9.1.1		
Pre-test conditions:	• Server accepts creation of new resource on uri **/location-query**, the location of the created resource contains two query parameters **?first=1&second=2**		
Test Sequence:	**Step**	**Type**	**Description**
	1	Stimulus	Client is requested to send a confirmable POST request to server's resource
	2	Check	The request sent by the client contains: • Type = 0 (CON) • Code = 2 (POST) • An arbitrary payload • Content-format option
	3	Check	Server sends response containing: • Code = 65 (2.01 created) • Two options whose type is Location-Query ■ The first option contains **first=1** ■ The second option contains **second=2**
	4	Verify	Client displays the response

Interoperability Test Description			
Identifier:	TD_COAP_CORE_20		
Objective:	Perform GET transaction containing the Accept option (CON mode)		
Configuration:	CoAP_CFG_01		
References:	[1] clause 5.8.1,5.10.5,5.10.4		
Pre-test conditions:	• Server should provide a resource **/multi-format** which exists in two formats: - text/plain;charset=utf-8 - application/xml		
Test Sequence:	**Step**	**Type**	**Description**
Part A: client requests a resource in text format			
	1	Stimulus	Client is requested to send a confirmable GET request to server's resource
	2	Check	The request sent request by the client contains: • Type = 0 (CON) • Code = 1 (GET) • Option: type = Accept, value = 0 (text/plain;charset=utf-8)
	3	Check	Server sends response containing: • Code = 69 (2.05 content) • Option type = Content-Format, value = 0 (text/plain;charset=utf-8) • Payload = Content of the requested resource in text/plain;charset=utf-8 format
	4	Verify	Client displays the response
Part B: client requests a resource in xml format			
	5	Stimulus	Client is requested to send a confirmable GET request to server's resource
	6	Check	The request sent by the client contains: • Type = 0 (CON) • Code = 1 (GET) • Option: type = Accept, value = 41 (application/xml)
	7	Check	Server sends response containing: • Code = 69 (2.05 content) • Option: type = Content-Format, value = 41 (application/xml) Payload = Content of the requested resource in application/xml format
	8	Verify	Client displays the response

Interoperability Test Description			
Identifier:	TD_COAP_CORE_21		
Objective:	Perform GET transaction containing the ETag option (CON mode)		
Configuration:	CoAP_CFG_01		
References:	[1] clause 5.8.1, 5.10.7,5.10.10,12.1.12		
Pre-test conditions:	• Server should offer a **/validate** resource which vary in time • Client & server supports ETag option • The Client 's cache must be purged		
Test Sequence:	**Step**	**Type**	**Description**
Part A: Verifying that client cache is empty			
	1	Stimulus	Client is requested to send a confirmable GET request to server's resource
	2	Check	The request sent request by the client contains: • Type = 0 (CON) • Code = 1 (GET) • No ETag option

Interoperability Test Description			
	3	Check	Server sends response containing: • Code = 69 (2.05 content) • Option type = ETag • Option value = an arbitrary ETag value Not empty Payload
	4	Verify	Client displays the response
Part B: Verifying client cache entry is still valid			
	5	Stimulus	Client is requested to send s confirmable GET request to server's resource so as to check if the resource was updated
	6	Check	The request sent by the client contains: • Type = 0 (CON) • Code = 1 (GET) • Option Type=ETag • Option value=the ETag value received in step 3
	7	Check	Server sends response containing: • Code = 67 (2.03 Valid) • Option type = ETag • Option value = the ETag value sent in step 3 • An empty payload
	8	Verify	Client displays the response
Part C: Verifying that client cache entry is no longer valid			
	9	Stimulus	Update the content of the server's resource from a CoAP client
	10	Stimulus	Client is requested to send a confirmable GET request to server's resource so as to check if the resource was updated
	11	Check	The request sent by the client contains: • Type = 0 (CON) • Code = 1 (GET) • Option Type=ETag Option value=the ETag value received in step 3
	12	Check	Server sends response containing: • Code = 69 (2.05 Content) • Option type = ETag • Option value = an arbitrary ETag value which differs from the ETag sent in step 3 • The payload of the requested resource, which should be different from the payload in step 3
	13	Verify	Client displays the response

Interoperability Test Description			
Identifier:	TD_COAP_CORE_22		
Objective:	Perform GET transaction with responses containing the ETag option and requests containing the If-Match option (CON mode)		
Configuration:	CoAP_CFG_01		
References:	[1] clause 5.8.1, 5.10.7,5.10.9,12.1.12		
Pre-test conditions:	• Server should offer a **/validate** resource • Client & server supports ETag and If-Match option • The Client 's cache must be purged		
Test Sequence:	**Step**	**Type**	**Description**
Preamble: client gets the resource			
	1	Stimulus	Client is requested to send a confirmable GET request to server's resource
	2	Check	The request sent by the client contains: • Type = 0 (CON) • Code = 1 (GET)

Interoperability Test Description			
	3	Check	Server sends response containing: • Code = 69 (2.05 content) • Option type = ETag • Option value = an arbitrary Etag value • Not empty Payload
Part A: single update			
	4	Stimulus	Client is requested to send a confirmable PUT request to server's resource so as to perform an atomic update
	5	Check	The request sent by the client contains: • Type = 0 (CON) • Code = 3 (PUT) • Option Type=If-Match • Option value=ETag value received in step 3 • An arbitrary payload (which differs from the payload received in step 3)
	6	Check	Server sends response containing: • Code = 68 (2.04 Changed) •
	7	Verify	Client displays the response and the server changed its resource
Part B: concurrent updates			
	8	Stimulus	Client is requested to send a confirmable GET request to server's resource
	9	Check	The request sent by the client contains: • Type = 0 (CON) • Code = 1 (GET)
	10	Check	Server sends response containing: • Code = 69 (2.05 content) • Option type = ETag • Option value = an arbitrary Etag value which differs from the ETag sent in step 3 • The Payload sent in step 5
	11	Verify	Client displays the response
	12	Stimulus	Update the content of the server's resource from a CoAP client
	13	Stimulus	Client is requested to send a confirmable PUT request to server's resource so as to perform an atomic update
	14	Check	The request sent by the client contains: • Type = 0 (CON) • Code = 3 (PUT) • Option Type=If-Match • Option value=ETag value received in step 10 • An arbitrary payload (which differs from the previous payloads)
	15	Check	Server sends response containing: • Code = 140 (4.12 Precondition Failed)
	16	Verify	Client displays the response and the server did not update the content of the resource

Interoperability Test Description			
Identifier:	TD_COAP_CORE_23		
Objective:	Perform PUT transaction containing the If-None-Match option (CON mode)		
Configuration:	CoAP_CFG_01		
References:	[1] clause 5.8.1, 5.10.7,5.10.10,12.1.12		
Pre-test conditions:	• Server should offer a **/create1** resource, which does not exist and which can be created by the client • Client & server supports If-Non-Match		
Test Sequence:	**Step**	**Type**	**Description**

Interoperability Test Description			
Part A: single creation			
	1	Stimulus	Client is requested to send a confirmable PUT request to server's resource so as to atomically create the resource.
	2	Check	The request sent by the client contains: • Type = 0 (CON) • Code = 3 (PUT) • Option Type=If-None-Match • An arbitrary payload
	3	Check	Server sends response containing: • Code = 65 (2.01 Created)
	4	Verify	Client displays the response and the server created a new resource
Part B: concurrent creations			
	5	Stimulus	Client is requested to send a confirmable PUT request to server's resource so as to atomically create the resource.
	6	Check	The request sent by the client contains: • Type = 0 (CON) • Code = 3 (PUT) • Option Type=If-None-Match • An arbitrary payload
	7	Check	Server sends response containing: • 140 (4.12 Precondition Failed)
	8	Verify	Client displays the response

Interoperability Test Description			
Identifier:	TD_COAP_CORE_24		
Objective:	Perform POST transaction with responses containing several Location-Path options (Reverse Proxy in CON mode)		
Configuration:	CoAP_CFG_03		
References:	[1] clause 5.8.1,5.10.8,5.9.1.1, 8.2.2,8.2.1,10.2.2,11.2		
Pre-test conditions:	• Proxy is configured as a reverse-proxy for the server • Proxy's cache is cleared • Server accepts creation of new resource on **/create2** and the created resource is located at **/location1/location2/location3** (resource does not exist yet)		
Test Sequence:	**Step**	**Type**	**Description**
	1	Stimulus	Client is requested to send a confirmable POST request to proxy
	2	Check	The POST sent by the client contains: • Type = 0 (CON) • Code = 2 (POST) • An arbitrary payload • Content-format option
	3	Check	The Proxy forwards the POST request to server's resource

Interoperability Test Description			
			and that it contains: • Type = 0 (CON) • Code = 2 (POST) • An arbitrary payload • Content-format option
	4	Check	Server sends a response to the proxy containing: • Code = 65 (2.01 created) • Option type = Location-Path (one for each segment) • Option values must contain “location1”, “location2” & “location3” without contain any ‘/’
	5	Check/	Observe that the Proxy forwards the response (in step 4) to client and check that the forwarded response contains: • Code = 65 (2.01 created) • Option type = Location-Path (one for each segment) • Option values must contain “location1”, “location2” & “location3” without contain any ‘/’
	6	Verify	Client displays the response
	7	Verify	Client interface returns the response • 2.01 created • Location: coap://*proxy*/location1/location2/location3

Interoperability Test Description			
Identifier:	TD_COAP_CORE_25		
Objective:	Perform POST transaction with responses containing several Location- Query option (Reverse proxy)		
Configuration:	CoAP_CFG_03		
References:	[1] clause 5.8.1,5.10.8,5.9.1.1, 8.2.2,8.2.1,10.2.2,11.2		
Pre-test conditions:	• Proxy is configured as a reverse-proxy for the server • Proxy’s cache is cleared • Server accepts creation of new resource on uri **/location-query**, the location of the created resource contains two query parameters **?first=1&second=2**		
Test Sequence:	**Step**	**Type**	**Description**
	1	Stimulus	Client is requested to send a confirmable POST request to proxy
	2	Check	Proxy receives the request from client & forwards it to server’s resource
	3	Check	Forwarded request must contain: • Type = 0 (CON) • Code = 2 (POST) • An arbitrary payload • Content-format option
	4	Check	Server sends response to proxy containing: • Code = 65 (2.01 created) • Two options whose type is Location-Query ■ The first option contains **first=1** ■ The second option contains **second=2**
	5	Check	Proxy forwards the response to client
	6	Check	Client displays the message
	7	Verify	Client interface returns the response: • 2.01 created • Location: coap://*proxy*/?first=1&second=2

Interoperability Test Description			
Identifier:	TD_COAP_CORE_26		
Objective:	Perform GET transaction containing the Accept option (CON mode		
Configuration:	CoAP_CFG_03		
References:	[1] clause 5.8.1,5.10.5,5.10.4, 8.2.2,8.2.1,10.2.2,11.2		
Pre-test conditions:	• Proxy is configured as a reverse-proxy for the server • Proxy's cache is cleared • Server should provide a resource **/multi-format** which exists in two formats: - text/plain;charset=utf-8 - application/xml		
Test Sequence:	**Step**	**Type**	**Description**
Part A: client requests text format			
	1	Stimulus	Client is requested to send a confirmable GET request to proxy
	2	Check	Proxy receives the request from client & forwards it to server's resource
	3	Check	Forwarded request must contain: • Type = 0 (CON) • Code = 1 (GET) • Option: type = Accept, value = 0 (text/plain;charset=utf-8)
	4	Check	Server sends response containing: • Code = 69 (2.05 content) • Option: type = Content-Format, value = 0 (text/plain;charset=utf-8) • Payload = Content of the requested resource in text/plain;charset=utf-8 format
	5	Check	Proxy forwards the response to client
	6	Verify	Client receives & displays the response
	7	Check	Response contains: • Code = 69 (2.05 content) • Option: type = Content-Format, value = 0 (text/plain;charset=utf-8) • Payload = Content of the requested resource in text/plain;charset=utf-8 format
Part B: client requests xml format			
	8	Stimulus	Client is requested to send a confirmable GET request to Proxy
	9	Check	Proxy forwards the request to server
	10	Check	Sent request must contain: • Type = 0 (CON) • Code = 1 (GET) Option: type = Accept, value = 41 (application/xml)
	11	Check	Server sends response containing: • Code = 69 (2.05 content) • Option: type = Content-Format, value = 41 (application/xml) Payload = Content of the requested resource in application/xml format
	12	Check	Proxy forwards the response to client
	13	Verify	Client receives & displays the response
	14	Check	Client displays the response received: • Code = 69 (2.05 content) • Option: type = Content-Format, value = 41 (application/xml) Payload = Content of the requested resource in application/xml format

Interoperability Test Description			
Identifier:	TD_COAP_CORE_27		
Objective:	Perform GET transaction with responses containing the ETag option and requests containing the If-Match option (CON mode)		
Configuration:	CoAP_CFG_03		
References:	[1] clause 5.8.1, 5.10.7,5.10.9,12.1.12, 8.2.2,8.2.1,10.2.2,11.2		
Pre-test conditions:	• Proxy is configured as a reverse-proxy for the server • Proxy's cache is cleared • Server should offer a /validate resource with resource content is not empty • Client & server supports ETag option and If-Match option		
Test Sequence:	**Step**	**Type**	**Description**
Preamble: client gets the resource			
	1	Stimulus	Client is requested to send a confirmable GET request to proxy
	2	Check	Proxy forwards the request to server
	3	Check	Forwarded request must contain: • Type = 0 (CON) • Code = 1 (GET)
	4	Check	Server sends response containing: • Code = 69 (2.05 content) • Option type = ETag • Option value = an arbitrary ETag value • Not empty payload
	5	Check	Proxy forwards the response to client
Part A: single update			
	6	Stimulus	Client is requested to send a confirmable PUT request to Proxy
	7	Check	Sent request must contain: • Type = 0 (CON) • Code = 3 (PUT) • Option Type=If-Match • Option value=ETag value received in step 4 • An arbitrary payload (which differs from the payload received in step 3)
	8	Verify	Proxy forwards the request to servers resource & server updates the resource
	9	Check	Server sends response containing: • Code = 68 (2.04 Changed) • Option type = ETag • Option value = an arbitrary ETag value which differs from the ETag received in step 4
	10	Check	Proxy forwards the response to client
	11	Check	Forwarded response contains: • Code = 68 (2.04 Changed) • Option type = ETag • Option value = same ETag value found in step 8
	12	Verify	Client displays the response
Part B: concurrent updates			
	13	Stimulus	Update the content of the server's resource from a CoAP client
	14	Stimulus	Client is requested to send s confirmable PUT request to proxy so as to perform an atomic update
	15	Check	Sent request must contain: • Type = 0 (CON) • Code = 3 (PUT) • Option Type=If-Match • Option value=ETag value received in step 8 An arbitrary payload (which differs from the previous payloads)
	16	Check	Proxy forwards the request to server's resource
	17	Check	Sent request must contain:

Interoperability Test Description			
			• Type = 0 (CON) • Code = 3 (PUT) • Option Type=If-Match • Option value=same ETag value found in step 14 An arbitrary payload (which differs from the previous payloads)
	18	Check	Server sends response containing: • Code = 140 (4.12 Precondition Failed)
	19	Verify	Proxy forwards the response to client
	20	Check	Response contains: Code = 140 (4.12 Precondition Failed)
	21	Verify	Client displays the response

Interoperability Test Description			
Identifier:	TD_COAP_CORE_28		
Objective:	Perform GET transaction with responses containing the ETag option and requests containing the If-None-Match option (CON mode) (Reverse proxy)		
Configuration:	CoAP_CFG_03		
References:	[1] clause 5.8.1, 5.10.7,5.10.10,12.1.12, 8.2.2,8.2.1,10.2.2,11.2		
Pre-test conditions:	• Proxy is configured as a reverse-proxy for the server • Proxy’s cache is cleared • Server should offer a /create3 resource, which does not exist and which can be created by the client • Client & server supports If-None-Match		
Test Sequence:	**Step**	**Type**	**Description**
Part A: single creation			
	1	Stimulus	Client is requested to send a confirmable PUT request to proxy to atomically create resource in server
	2	Check	Proxy forwards the request to server
	3	Check	Forwarded t request must contain: • Type = 0 (CON) • Code = 3 (PUT) • Option Type=If-None-Match • An arbitrary payload
	4	Check	Server sends response containing: • Code = 65 (2.01 Created)
	5	Check	Proxy forwards the response to client
	6	Verify	Client displays the response & and server created new resource
Part B: concurrent creations			
	5	Stimulus	Client is requested to send s confirmable PUT request to proxy to atomically create resource in server
	6	Check	Sent request must contain: • Type = 0 (CON) • Code = 3 (PUT) • Option Type=If-Non-Match • Option value=Received ETag value
	7	Check	Server sends response containing: • 140 (4.12 Precondition Failed)
	8	Verify	Proxy forwards the response to client
	9	Check	Response contains: • 140 (4.12 Precondition Failed)

Interoperability Test Description			
	10	Verify	Client displays the response

Interoperability Test Description			
Identifier:	TD_COAP_CORE_29		
Objective:	Perform GET transaction with responses containing the Max-Age option (Reverse proxy)		
Configuration:	CoAP_CFG_03		
References:	[1] clause 5.8.1,5.10.6,5.9.1.3,5.9.1.5, 8.2.2,8.2.1,10.2.2,11.2		
Pre-test conditions:	• Proxy offers a cache • Proxy is configured as a reverse-proxy for the server • Servers resource vary in time and supports Max-Age option • Proxy's cache is cleared • Server offers a resource /validate that varies in time, with a Max-Age set to 30s		
Test Sequence:	**Step**	**Type**	**Description**
	1	Stimulus	A confirmable GET request is sent to Proxy from Client
	2	Check	Proxy Sends request containing: • Type = 0 (CON) • Code = 1 (GET)
	3	Check	Server sends response containing: • Code = 69 (2.05 Content) • Option type = ETag • Option value = ETag value • Option type = Max-age • Option value • Not empty Payload
	4	Verify	Proxy forwards response to client
	5	Stimulus	A confirmable GET request is sent to proxy from Client before Max-Age expires
	6	Check	Proxy dos not forward any request to the server
	7	Check	Proxy sends response to client
	8	Verify	Response contains: • Option type = Max-age • Option Value = new Max-age • Payload cached

7.2 CoRE Link Format

Interoperability Test Description			
Identifier:	TD_COAP_LINK_01		
Objective:	Access to well-known interface for resource discovery		
Configuration:	CoAP_CFG_01		
References:	[2]		
Pre-test conditions:	• Client and server supports CoRE Link Format • Server supports **/.well-known/core** resource and the CoRE Link Format		
Test Sequence:	**Step**	**Type**	**Description**
	1	Stimulus	Client is requested to retrieve Server's list of resource
	2	Check	Client sends a GET request to Server for /.well-known/core resource
	3	Check	Server sends response containing: Content-format option indicating 40 (application/link-format) Code indicating 69 (2.05 content) Payload indicating all the links available on Server
	4	Verify	Client displays the list of resources available on Server

Interoperability Test Description			
Identifier:	TD_COAP_LINK_02		
Objective:	Use filtered requests for limiting discovery results		
Configuration:	CoAP_CFG_01		
References:	[2] 4.1		
Pre-test conditions:	• Client supports CoRE Link Format • Server supports CoRE Link Format • Server offers different types of resources (*Type1*, *Type2*, ...; see Note)		
Test Sequence:	**Step**	**Type**	**Description**
	1	Stimulus	Client is requested to retrieve Server's list of resource of a specific type *Type1*
	2	Check	Client sends a GET request to Server for /.well-known/core resource containing URI-Query indicating "rt=*Type1*"
	3	Check	Server sends response containing: Content- format option indicating 40 (application/link-format) Payload indicating only the links of type *Type1* available on Server
	4	Verify	Client displays the list of resources of type *Type1* available on Server
Note: *Type1*, *Type2*, ... refer to real resource types available on Server and shall be extracted from Server's **/.well-known/core** resource			

Interoperability Test Description			
Identifier:	TD_COAP_LINK_03		
Objective:	Handle empty prefix value strings		
Configuration:	CoAP_CFG_01		
References:	[2] 4.1 §2		
Pre-test conditions:	• Client supports Core Link Format • Server supports Core Link Format • Server offers different types of resources (*Type1*, *Type2*, ...; see Note) • Server offers resources with no type (i.e. no rt attribute)		
Test Sequence:	**Step**	**Type**	**Description**
	1	Stimulus	Client is requested to retrieve Server's list of resources matching an rt empty value
	2	Check	Client sends a GET request to Server for /.well-known/core resource containing URI-Query indicating rt="*"
	3	Check	Server sends response containing: Content-format option indicating 40 (application/link-format) Payload indicating only the links having an rt attribute
	4	Verify	Client displays the list of resources with rt attribute available on Server
Note: *Type1*, *Type2*, ... refer to real resource types available on Server and shall be extracted from Server's **/.well-known/core** resource			

Interoperability Test Description			
Identifier:	TD_COAP_LINK_04		
Objective:	Filter discovery results in presence of multiple rt attributes		
Configuration:	CoAP_CFG_01		
References:	[2] 3.1, 4.1 §2		
Pre-test conditions:	• Client supports Core Link Format • Server supports Core Link Format • Server offers 4 groups of resources: 1. Resources with rt="Type1 Type2" 2. Resources with rt="Type2 Type3" 3. Resources with rt="Type1 Type3" 4. Resources with rt=""		
Test Sequence:	**Step**	**Type**	**Description**
	1	Stimulus	Client is requested to retrieve Server's list of resources of a specific type *Type2*
	2	Check	Client sends a GET request to Server for /.well-known/core resource containing URI-Query indicating rt="Type2"
	3	Check	Server sends response containing: Content-format option indicating 40 (application/link-format) Payload indicating only the links of groups 1 and 2
	4	Verify	Client displays the list of resources of type *Type2* available on Server

Interoperability Test Description			
Identifier:	TD_COAP_LINK_05		
Objective:	Filter discovery results using if attribute and prefix value strings		
Configuration:	CoAP_CFG_01		
References:	[2] 3.2, 4.1 §5		
Pre-test conditions:	• Client supports Core Link Format • Server supports Core Link Format • Server offers 4 groups of resources: 1. Resources with if="If1" 2. Resources with if="If2" 3. Resources with if="foo" 4. Resources with no if attribute		
Test Sequence:	**Step**	**Type**	**Description**
	1	Stimulus	Client is requested to retrieve Server's list of resources matching the interface description pattern "If*"
	2	Check	Client sends a GET request to Server for /.well-known/core resource containing URI-Query indicating if="If*"
	3	Check	Server sends response containing: Content-format option indicating 40 (application/link-format) Payload indicating only the links of groups 1 and 2
	4	Verify	Client displays the retrieved list of resources

Interoperability Test Description			
Identifier:	TD_COAP_LINK_06		
Objective:	Filter discovery results using sz attribute and prefix value strings		
Configuration:	CoAP_CFG_01		
References:	[2] 3.3, 4.1 §5		
Pre-test conditions:	• Client supports Core Link Format • Server supports Core Link Format • Server offers resource with sz attribute • Server offers resources with no sz attribute		
Test Sequence:	**Step**	**Type**	**Description**
	1	Stimulus	Client is requested to retrieve Server's list of resources having a sz attribute
	2	Check	Client sends a GET request to Server for /.well-known/core resource containing URI-Query indicating sz="*"
	3	Check	Server sends response containing: Content-format option indicating 40 (application/link-format) Payload indicating only the links having a sz attribute
	4	Verify	Client displays the retrieved list of resources

Interoperability Test Description			
Identifier:	TD_COAP_LINK_07		
Objective:	Filter discovery results using href attribute and complete value strings		
Configuration:	CoAP_CFG_01		
References:	[2] 4.1		
Pre-test conditions:	• Client supports Core Link Format • Server supports Core Link Format • Server offers resources /link1 /link2 and /link3		
Test Sequence:	**Step**	**Type**	**Description**
	1	Stimulus	Client is requested to retrieve the link-value anchored at /link1
	2	Check	Client sends a GET request to Server for /.well-known/core resource containing URI-Query indicating href="/link1"
	3	Check	Server sends response containing: Content-format option indicating 40 (application/link-format) Payload indicating only the link for /link1
	4	Verify	Client displays the retrieved list of resources

Interoperability Test Description			
Identifier:	TD_COAP_LINK_08		
Objective:	Filter discovery results using href attribute and prefix value strings		
Configuration:	CoAP_CFG_01		
References:	[2] 4.1		
Pre-test conditions:	• Client supports Core Link Format • Server supports Core Link Format • Server offers resources /link1 /link2 and /link3 • Server offers resource /test		
Test Sequence:	**Step**	**Type**	**Description**
	1	Stimulus	Client is requested to retrieve the link-value anchored at /link*
	2	Check	Client sends a GET request to Server for /.well-known/core resource containing URI-Query indicating href="/link*"
	3	Check	Server sends response containing: Content-format option indicating 40 (application/link-format) Payload indicating only the link matching /link*
	4	Verify	Client displays the retrieved list of resources

Interoperability Test Description			
Identifier:	TD_COAP_LINK_09		
Objective:	Arrange link descriptions hierarchically		
Configuration:	CoAP_CFG_01		
References:	[2] 5 §4		
Pre-test conditions:	• Client supports Core Link Format • Server supports Core Link Format • Server offers an entry located at /path with ct=40 • Server offers sub-resources /path/sub1, /path/sub2, … (see Note)		
Test Sequence:	**Step**	**Type**	**Description**
	1	Stimulus	Client is requested to retrieve one of the sub-resources
	2	Check	Client sends a GET request to Server for /.well-known/core resource
	3	Check	Server sends response containing: Content-format option indicating 40 (application/link-format) Payload indicating the link description for /path
	4	Check	Client sends a GET request for /path to Server
	5	Check	Server sends response containing: Content-format option indicating 40 (application/link-format) Payload indicating the link description for /path/sub1, /path/sub2, …
	6	Check	Client sends a GET request for /path/sub1
	7	Check	Server sends 2.05 (Content) response. Payload contains /path/sub1
	8	Verify	Client displays the retrieved sub-resource.
Note: /path/sub1, /path/sub2, … refer to real resources available on Server and shall be extracted from Server's **/.well-known/core** resource			

7.3 Blockwise transfers

Identifier:	TD_COAP_BLOCK_01		
Objective:	Handle GET blockwise transfer for large resource (early negotiation)		
Configuration:	CoAP_CFG_01		
References:	[4] 2.2		
Pre-test conditions:	• Client supports Block transfers • Server supports Block transfers • Server offers a large resource **/large** • Client knows /large requires block transfer		
Test Sequence:	**Step**	**Type**	**Description**
	1	Stimulus	Client is requested to retrieve resource /large
	2	Check	Client sends a GET request. The request optionally contains a Block2 option indicating: • NUM = 0; • M = 0; • SZX = the desired block size.
	3	Check	Server sends 2.05 (Content) response with a Block2 option indicating: • NUM = 0; • M = 1; • SZX is less or equal to the desired block size indicated by the GET request. Payload size is 2^{SZX+4} bytes.
	4*	Check	Client send GET requests for further blocks indicating: • NUM = i where "i" is the block number of the current block; • M = 0; • SZX is the SZX at step 3.
	5*	Check	Server sends 2.05 (Content) response containing Block2 option indicating: • NUM = i where "i" is the block number used at step 4; • M = 1; • SZX is the SZX at step 3. Payload size MUST be 2^{SZX+4} bytes.
	6	Check	Client send GET request for the last block indicating: • NUM = n where "n" is the last block number; • M = 0; • SZX is the SZX at step 3.
	7	check	Server sends 2.05 (Content) response with a Block2 option indicating: • NUM = n where "n" is the block number used at step 6; • M = 0; • SZX is the SZX at step 3. Payload size is lesser or equal to 2^{SZX+4} bytes.
	8	Verify	Client displays the received information
(*)Note: Steps 4 and 5 are in a loop.			

Identifier:	TD_COAP_BLOCK_02		
Objective:	Handle GET blockwise transfer for large resource (late negotiation)		
Configuration:	CoAP_CFG_01		
References:	[4] 2.2		
Pre-test conditions:	• Client supports Block transfers • Server supports Block transfers • Server offers a large resource **/large** • Client does not know /large requires block transfer		
Test Sequence:	**Step**	**Type**	**Description**
	1	Stimulus	Client is requested to retrieve resource /large
	2	Check	Client sends a GET request not containing Block2 option
	3	Check	Server sends 2.05 (Content) response with a Block2 option indicating: • NUM = 0; • M = 1; • SZX = the proposed block size. Payload size is 2^{SZX+4} bytes.
	4	Check	Client switches to blockwise transfer mode and sends a GET request with a Block2 option indicating: • NUM is the next block number (should be equal to $2^{SZX\_in\_step\_4 - SZX\_in\_step\_3}$); • M = 0; • SZX is less or equals to SZX at step 3.
	5	Check	Server sends 2.05 (Content) response with a Block2 option indicating: • NUM = k where “k” is the block number used at step 4; • M = 1; • SZX is the SZX at step 4. Payload size is 2^{SZX+4} bytes.
	6*	Check	Client sends GET request for further blocks indicating: • NUM = i where “i” is the block number of the current block; • M = 0; • SZX is the SZX at step 4.
	7*	Check	Server sends 2.05 (Content) response with a Block2 option indicating: • NUM = i where “i” is the block number used at step 6; • M = 1; • SZX is the SZX at step 4. Payload size is 2^{SZX+4} bytes.
	8	Check	Client send GET request for the last block indicating: • NUM = n where “n” is the last block number; • M = 0; SZX is the SZX at step 4.
	9	Check	Server sends 2.05 (Content) response with a Block2 option indicating: • NUM = n where “n” is the block number used at step 8; • M = 0; • SZX is the SZX at step 4. Payload size is lesser or equal to 2^{SZX+4}.
	10	Verify	Client displays the received information
(*) Note: Steps 6 and 7 are in a loop.			

Identifier:	TD_COAP_BLOCK_03		
Objective:	Handle PUT blockwise transfer for large resource		
Configuration:	CoAP_CFG_01		
References:	[4] 2.2		
Pre-test conditions:	• Client supports Block transfers • Server supports Block transfers • Server offers a large updatable resource **/large-update**		
Test Sequence:	**Step**	**Type**	**Description**
	1	Stimulus	Client is requested to update resource /large-update on Server
	2	check	Client sends a PUT request containing Block1 option indicating: • NUM = 0; • M = 1; • SZX = the desired block size. Payload size is 2^{SZX+4} bytes.
	3	Check	Server sends 2.04 (Changed) response with a Block1 option indicating: • NUM = 0; • M = 0 (stateless) or 1 (atomic); • SZX is less or equal to the SZX at step 2.
	4*	Check	Client sends further requests containing Block1 option indicating: • NUM = i where “i” is the block number of the current block. If the server decreased the SZX parameter in step 3, then the client should adapt the block size accordingly and may resume the transfer from block id $2^{size\_in\_step\_2-size\_in\_step\_3}$ instead of block 1) • M = 1; • SZX is the SZX at step 3. Payload size is 2^{SZX+4} bytes.
	5*	Check	Server sends 2.04 (Changed) response containing Block1 option indicating: • NUM = i where “i” is the block number used at step 4; • M = 0 (stateless) or 1 (atomic); • SZX is the SZX at step 3.
	6	Check	Client send PUT request containing the last block and indicating: • NUM = n where “n” is the last block number; • M = 0; • SZX is the SZX at step 3. Payload size is lesser or equal to 2^{SZX+4}.
	7	Check	Server sends 2.04 (Changed) response with a Block1 option indicating: • NUM = n where “n” is the block number used at step 6; • M = 0; • SZX is the SZX at step 3.
	8	Verify	Server indicates presence of the complete updated resource /large-update
(*) Note: Steps 4 and 5 are in a loop.			

Identifier:	TD_COAP_BLOCK_04		
Objective:	Handle POST blockwise transfer for large resource		
Configuration:	CoAP_CFG_01		
References:	[4] 2.2		
Pre-test conditions:	• Client supports Block transfers • Server supports Block transfers • Server accepts creation of new resources on **/large-create**		
Test Sequence:	**Step**	**Type**	**Description**
	1	Stimulus	Client is requested to create a new resource /large-create on Server

	2	Check	Client sends a POST request containing Block1 option indicating: • NUM = 0; • M = 1; • SZX = the desired block size. Payload size is 2^{SZX+4} bytes.
	3	Check	Server sends 2.01 (Created) response containing Block1 option indicating: • NUM = 0; • M = 0 (stateless) or 1 (atomic); • SZX is less or equal to the SZX at step 2.
	4*	Check	Client sends further requests containing Block1 option indicating: • NUM = i where "i" is the block number of the current block. If the server decreased the SZX parameter in step 3, then the client should adapt the block size accordingly and may resume the transfer from block id $2^{size\_in\_step\_2-size\_in\_step\_3}$ instead of block 1) • M = 1; • SZX is the SZX at step 3. Payload size is 2^{SZX+4} bytes.
	5*	Check	Server sends 2.01 (Created) response containing Block1 option indicating: • NUM = i where "i" is the block number used at step 4; • M = 1; • SZX is the SZX at step 3
	6	Check	Client send PUT request containing the last block and indicating: • NUM = n where "n" is the last block number; • M = 0; • SZX is the SZX at step 3. Payload size is lesser or equal to 2^{SZX+4}.
	7	Check	Server sends 2.01 (Created) response containing Block1 option indicating: • NUM = n where "n" is the block number used at step 6; • M = 0; • SZX is the SZX at step 3.
	8	Verify	Server indicates presence of the complete new resource /large-create
(*) Note: Steps 4 and 5 in a loop.			

7.4 Observing Resources

Interoperability Test Description			
Identifier:	TD_COAP_OBS_01		
Objective:	Handle resource observation with CON messages		
Configuration:	CoAP_CFG_01		
References:	[3] 1.2,		
Pre-test conditions:	• Client supports Observe option • Server supports Observe option • Server offers an observable resource **/obs** which changes periodically (e.g. every 5s) which produces confirmable notifications		
Test Sequence:	**Step**	**Type**	**Description**
	1	Stimulus	Client is requested to send to the server a confirmable GET request with observe option for resource /obs
	2	Check	The request sent by client contains: • Type = 0 (CON) • Code = 1 (GET) • Token value = a value generated by the client • Observe option = empty
	3	Check	Server sends the response containing: • Type = 2 (ACK) • Content-format of the resource /obs • Token value = same as one found in the step 2 • Observe option with a sequence number
	4[1]	Check	Server sends a notification containing: • Type = 0 (CON) • Content-format = same as one found in the step 3 • Token value = same as one found in the step 3 • Observe option indicating increasing values
	5	Verify	Client displays the received information
	6	Check	Client sends an ACK
Notes: **(1)** Steps 4-6 are in a loop.			

Interoperability Test Description			
Identifier:	TD_COAP_OBS_02		
Objective:	Handle resource observation with NON messages		
Configuration:	CoAP_CFG_01		
References:	[3] 1.2,		
Pre-test conditions:	• Client supports Observe option • Server supports Observe option • Server offers an observable resource **/obs-non** which changes periodically (e.g. every 5s) which produces non-confirmable notifications		
Test Sequence:	**Step**	**Type**	**Description**
	1	Stimulus	Client is requested to send to the server a non-confirmable GET request with observe option for resource /obs
	2	Check	The request sent by client contains: • Type = 1 (NON) • Code = 1 (GET) • Token value = a value generated by the client • Observe option = empty
	3[1]	Check	Server sends a notification containing: • Type = 1 (NON) • Content-format = the same for all notifications

Interoperability Test Description			
			• Token value = same as one found in the step 2 • Observe option indicating increasing values
	4	Verify	Client displays the received information
Notes: **(1)** Steps 3- 4 are in a loop.			

Interoperability Test Description			
Identifier:	TD_COAP_OBS_03		
Objective:	Stop resource observation		
Configuration:	CoAP_CFG_01		
References:	[3] 4.1 §3		
Pre-test conditions:	• Client supports Observe option • Server supports Observe option • Server offers an observable resource **/obs** which changes periodically (e.g. every 5s) which produces confirmable notifications		
Test Sequence:	**Step**	**Type**	**Description**
	1	Stimulus	Client is requested to send to the server a confirmable GET request with observe option for resource /obs
	2	Check	The request sent by client contains: • Type = 0 (CON) • Code = 1 (GET) • Token value = a value generated by the client • Observe option = empty
	3	Check	Server sends the response containing: • Type = 2 (ACK) • Content-format of the resource /obs • Token value = same as one found in the step 2 • Observe option with a sequence number
	4[1]	Check	Server sends a notification containing: • Type = 0 (CON) • Content-format = same as one found in the step 3 • Token value = same as one found in the step 2 • Observe option indicating increasing values
	5	Check	Client displays the received information
	6	Check	Client sends an ACK
	7[2]	Stimulus	Client is requested to stop observing the resource /obs on the server
	8	Check	Client sends a request containing : • Type = 0 (CON) • Code = 1 (GET) • Token value = a value generated by the client • DOES NOT contain observe option
	9	Check	Server sends response not containing Observe option
	10	Verify	Client displays the received information
	11	Check	Server does not send further response
	12	Verify	Client does not display updated information
Notes: **(1)** Steps 4-6 are in a loop. **(2)** Step 7-12 are asynchronous to the loop.			

Interoperability Test Description			
Identifier:	TD_COAP_OBS_04		
Objective:	Client detection of deregistration (Max-Age)		
Configuration:	CoAP_CFG_01		
References:	[3] 3.3 §4		
Pre-test conditions:	• Client supports Observe option • Server supports Observe option • Server offers an observable resource **/obs** which changes periodically (e.g. every 5s) which produces confirmable notifications		
Test Sequence:	**Step**	**Type**	**Description**
	1	Stimulus	Client is requested to send to the server a confirmable GET request with observe option for resource /obs
	2	Check	The request sent by client contains: • Type = 0 (CON) • Code = 1 (GET) • Token value = a value generated by the client • Observe option = empty
	3	Check	Server sends the response containing: • Type = 2 (ACK) • Content-format of the resource /obs • Token value = same as one found in the step 2 • Observe option with a sequence number
	4[1]	Check	Server sends a notification containing: • Type = 0 (CON) • Content-format = same as one found in the step 3 • Token value = same as one found in the step 2 • Observe option indicating increasing values
	5	Verify	Client displays the received information
	6	Check	Client sends an ACK
	7[2]	Stimulus	Server is rebooted
	8	Check	Server does not send notifications
	9	Verify	Client does not display updated information
	10	Verify	After Max-Age expiration[4] the client internally decides to send another GET request to the server with observe option for resource /obs
	11	Verify	Client sends a GET request to the server for resource /obs: • Type = 0 (CON) • Code = 1 (GET) • Token value = a value generated by the client different from the token at step 2 • Observe option = empty
	12	Check	Server sends the response containing: • Type = 2 (ACK) • Content-format of the resource /obs • Token value = same as one found in the step 11 • Observe option with a sequence number
	13[3]	Check	Server sends a notification containing: • Type = 0 (CON) • Content-format = same as one found in the step 12 • Token value = same as one found in the step 11 • Observe option indicating increasing values
	14	Verify	Client displays the received information
	15	Check	Client sends an ACK
Notes: **(1)** Steps 4-6 are in a loop. **(2)** Step 7-9 are asynchronous to the loop 4-6. **(3)** Steps 13-15 are in a loop. **(4)** A new registration should be attempted after Max-Age + MAX_LATENCY as recommended by [3]. MAX_LATENCY is defined by [1] and set to 100 seconds.			

Interoperability Test Description			
Identifier:	TD_COAP_OBS_05		
Objective:	Server detection of deregistration (client OFF)		
Configuration:	CoAP_CFG_01		
References:	[3] 4.5 §2		
Pre-test conditions:	• Client supports Observe option • Server supports Observe option • Server offers an observable resource **/obs** which changes periodically (e.g. every 5s) which produces confirmable notifications		
Test Sequence:	**Step**	**Type**	**Description**
	1	Stimulus	Client is requested to send to the server a confirmable GET request with observe option for resource /obs
	2	Check	The request sent by client contains: • Type = 0 (CON) • Code = 1 (GET) • Token value = a value generated by the client • Observe option = empty
	3	Check	Server sends the response containing: • Type = 2 (ACK) • Content-format of the resource /obs • Token value = same as one found in the step 2 • Observe option with a sequence number
	4[1]	Check	Server sends a notification containing: • Type = 0 (CON) • Content-format = same as one found in the step 3 • Token value = same as one found in the step 2 • Observe option indicating increasing values
	5	Check	Client displays the received information
	6	Check	Client sends an ACK
	7[2]	Stimulus	Client is switched off
	8	Check	Server's confirmable responses are not acknowledged Server's retransmissions have an updated Observe option value
	9	Check	Server should keep retransmitting the responses until at least Max-Age seconds after the first un-acknowledged response.

Notes:
(1) Steps 4-6 are in a loop.
(2) Step 7-12 are asynchronous to the loop.

Interoperability Test Description			
Identifier:	TD_COAP_OBS_06		
Objective:	Server detection of deregistration (explicit RST)		
Configuration:	CoAP_CFG_01		
References:	[3] 4.2 §5		
Pre-test conditions:	• Client supports Observe option • Server supports Observe option • Server offers an observable resource **/obs** which changes periodically (e.g. every 5s) which produces confirmable notifications		
Test Sequence:	**Step**	**Type**	**Description**
	1	Stimulus	Client is requested to send to the server a confirmable GET request with observe option for resource /obs
	2	Check	The request sent by client contains: • Type = 0 (CON) • Code = 1 (GET) • Token value = a value generated by the client • Observe option = empty
	3	Check	Server sends the response containing: • Type = 2 (ACK) • Content-format of the resource /obs • Token value = same as one found in the step 2 • Observe option with a sequence number
	4[1]	Check	Server sends a notification containing: • Type = 0 (CON) • Content-format = same as one found in the step 3 • Token value = same as one found in the step 2 • Observe option indicating increasing values
	5	Check	Client displays the received information
	6	Check	Client sends an ACK
	7[2]	Stimulus	Client is rebooted
	8	Check	Server is still sending notifications for the request in step 2. Notification contains: • Type = 0 (CON) • Content-format = same as one found in the step 3 • Token value = same as one found in the step 2 • Observe option indicating increasing values
	9	Verify	Client discards response and does not display information
	10	Check	Client sends RST to Server
	11	Verify	Server does not send further response
	12	Verify	Client does not display further received information
Notes: **(1)** Steps 4-6 are in a loop. **(2)** Step 7-12 are asynchronous to the loop.			

Interoperability Test Description			
Identifier:	TD_COAP_OBS_07		
Objective:	Server cleans the observers list on DELETE		
Configuration:	CoAP_CFG_01		
References:	[3] 3.2 §4		
Pre-test conditions:	• Client supports Observe option • Server supports Observe option • Server offers an observable resource **/obs** which changes periodically (e.g. every 5s) which produces confirmable notifications		
Test Sequence:	**Step**	**Type**	**Description**
	1	Stimulus	Client is requested to send to the server a confirmable GET request with observe option for resource /obs
	2	Check	The request sent by client contains: • Type = 0 (CON) • Code = 1 (GET) • Token value = a value generated by the client • Observe option = empty
	3	Check	Server sends the response containing: • Type = 2 (ACK) • Content-format of the resource /obs • Token value = same as one found in the step 2 • Observe option with a sequence number
	4[1]	Check	Server sends a notification containing: • Type = 0 (CON) • Content-format = same as one found in the step 3 • Token value = same as one found in the step 2 • Observe option indicating increasing values
	5	Check	Client displays the received information
	6	Check	Client sends an ACK
	7[2]	Stimulus	Delete the /obs resource of the server (either locally or by having another CoAP client perform a DELETE request)
	8[3]	Check	Server sends a notification containing: • Type = 0 (CON) • Code = 132 (4.04 NOT FOUND) • Token value = same as one found in the step 2 • Observe option indicating increasing values
	9	Verify	Server does not send further responses
	10	Verify	Client does not display further received information
Notes: **(1)** Steps 4-6 are in a loop. **(2)** Step 7-10 are asynchronous to the loop.			

Interoperability Test Description			
Identifier:	TD_COAP_OBS_08		
Objective:	Server cleans the observers list when observed resource content-format changes		
Configuration:	CoAP_CFG_01		
References:	[3] 4.2 §3		
Pre-test conditions:	• Client supports Observe option • Server supports Observe option • Server offers an observable resource **/obs** which changes periodically (e.g. every 5s) which produces confirmable notifications		
Test Sequence:	**Step**	**Type**	**Description**
	1	Stimulus	Client is requested to send to the server a confirmable GET request with observe option for resource /obs
	2	Check	The request sent by client contains: • Type = 0 (CON) • Code = 1 (GET) • Token value = a value generated by the client • Observe option = empty
	3	Check	Server sends the response containing: • Type = 2 (ACK) • Content-format of the resource /obs • Token value = same as one found in the step 2 Observe option with a sequence number
	4[1]	Check	Server sends a notification containing: • Type = 0 (CON) • Content-format = same as one found in the step 3 • Token value = same as one found in the step 2 • Observe option indicating increasing values
	5	Check	Client displays the received information
	6	Check	Client sends an ACK
	7[2]	Stimulus	Update the /obs resource of the server's resource with a new payload having a different Content-Format (either locally or by having another CoAP client perform a DELETE request)
	8[3]	Check	Server sends notification containing: • Type = 0 (CON) • Code = 160 (5.00 INTERNAL SERVER ERROR) • Token value = same as one found in the step 2 • Observe option indicating increasing values
	9	Verify	Server does not send further notifications
	10	Verify	Client does not display further received information
Notes: **(1)** Steps 4-6 are in a loop. **(2)** Step 7-10 are asynchronous to the loop.			

Interoperability Test Description			
Identifier:	TD_COAP_OBS_09		
Objective:	Update of the observed resource		
Configuration:	CoAP_CFG_01		
References:	[3] 4.2 §3		
Pre-test conditions:	• Client supports Observe option • Server supports Observe option • Server offers an observable resource **/obs** which changes periodically (e.g. every 5s) which produces confirmable notifications		
Test Sequence:	**Step**	**Type**	**Description**
	1	Stimulus	Client is requested to send to the server a confirmable GET request with observe option for resource /obs
	2	Check	The request sent by client contains: • Type = 0 (CON) • Code = 1 (GET) • Token value = a value generated by the client

			• Observe option = empty
	3	Check	Server sends the response containing: • Type = 2 (ACK) • Content-format of the resource /obs • Token value = same as one found in the step 2 • Observe option with a sequence number
	4[1]	Check	Server sends a notification containing: • Type = 0 (CON) • Content-format = same as one found in the step 3 • Token value = same as one found in the step 2 • Observe option indicating increasing values
	5	Check	Client displays the received information
	6	Check	Client sends an ACK
	7[2]	Stimulus	Update the /obs resource of the server’s resource with a new payload having the same Content-Format (either locally or by having another CoAP client perform a DELETE request)
	8[3]	Check	Server notifications contains: • Type = 0 (CON) • Content-format = same as one found in the step 3 • Token value = same as one found in the step 2 • Observe option indicating increasing values • Payload = the new value sent at step 8
	9	Verify	Client displays the new value of /obs sent in step 8
	10	Check	Client sends an ACK
Notes: **(1)** Steps 4-6 are in a loop. **(2)** Step 7-9 are asynchronous to the loop 4-6. **(3)** Steps 8-10 are in a loop (the same loop at steps 4-6 but /obs is updated).			

7.5 CoAP Binding for M2M REST Resources

7.5.1 ApplicationCreateRequest

Interoperability Test Description			
Identifier:	TD_M2M_COAP_01		
Objective:	M2M DA registers to its local SCL via an applicationCreateRequest (CoAP POST) and receives an applicationCreateResponse		
Configuration:	M2M_CFG_01		
References:	[5] 10.8.2, Annex D [**Erreur ! Source du renvoi introuvable.**] 9.3.2.8		
Pre-test conditions:	void		
Test Sequence:	**Step**	**Type**	**Description**
	1	Stimulus	M2M DA is requested to send a applicationCreateRequest (CoAP POST)
	2	Check (dIa)	Sent POST request contains • Code = 2(POST) • Uri-Path: <sclBase> • Uri-Path: applications • Payload: application resource <app> to be created • Content Format option = 41 (application/xml)
	3	Check (dIa)	SCL sends response containing: • Code = 65(2.01 Created) • Location-Path: <sclBase> • Location-Path: applications • Location-Path: <app> • The same Message ID as that of the previous request • Content Format option = 41 (application/xml) • Payload: applicationCreateResponse representation
	4	Verify (dIa)	M2M DA indicates successful operation

7.5.2 ApplicationRetrieveRequest

Interoperability Test Description			
Identifier:	TD_M2M_COAP_02		
Objective:	M2M DA retrieves application resource via an applicationRetrieveRequest (CoAP GET) and receives an applicationRetrieveResponse from its local SCL		
Configuration:	M2M_CFG_01		
References:	[5] 10.8.3, Annex D [**Erreur ! Source du renvoi introuvable.**] 9.3.2.8		
Pre-test conditions:	• DA has created an Application resource <app> on SCL		
Test Sequence:	**Step**	**Type**	**Description**
	1	Stimulus	M2M DA is requested to send a applicationRetrieveRequest (CoAP GET)
	2	Check (dIa)	Sent GET request contains • Code = 1(GET) • Uri-Path: <sclBase> • Uri-Path: applications • Uri-Path: <app> •
	3	Check (dIa)	SCL sends response containing: • Code = 69(2.05 Content) • The same Message ID as that of the previous request • Content Format option = 41 (application/xml) • Payload: application resource for <app> (applicationRetrieveResponse)
	4	Verify (dIa)	M2M DA indicates successful operation

7.5.3 ApplicationUpdateRequest

Interoperability Test Description			
Identifier:	TD_M2M_COAP_03		
Objective:	M2M DA updates attribute in application resource via an applicationUpdateRequest (CoAP PUT) and receives an applicationUpdateResponse from its local SCL		
Configuration:	M2M_CFG_01		
References:	[5] 10.8.4, Annex D [**Erreur ! Source du renvoi introuvable.**] 9.3.2.8		
Pre-test conditions:	• DA has created an Application resource <app> on SCL		
Test Sequence:	**Step**	**Type**	**Description**
	1	Stimulus	M2M DA is requested to send a applicationUpdateRequest (CoAP PUT)
	2	Check (dIa)	Sent PUT request contains • Code = 3 (PUT) • Uri-Path: <sclBase> • Uri-Path: applications • Uri-Path: <app> • Payload: modified application resource (e.g. modifies aPoc attribute) • Content Format option = 41 (application/xml)
	3	Check (dIa)	SCL sends response containing: • Code = 68 (2.04 Changed) • The same Message ID as that of the previous request • Content Format option = 41 (application/xml) • Payload: applicationUpdateResponse representation
	4	Verify (dIa)	M2M DA indicates successful operation

7.5.4 SubscriptionCreateRequest

Interoperability Test Description			
Identifier:	TD_M2M_COAP_04		
Objective:	M2M DA creates a subscription to application resource via subscriptionCreateRequest (CoAP POST) and receives a subscriptionCreateResponse from its local SCL		
Configuration:	M2M_CFG_01		
References:	[5] 10.25.2, Annex D [**Erreur ! Source du renvoi introuvable.**] 9.3.2.8.19		
Pre-test conditions:	• DA has created an Application resource <app> on SCL		
Test Sequence:	**Step**	**Type**	**Description**
	1	Stimulus	M2M DA is requested to send a subscriptionCreateRequest (CoAP POST)
	2	Check (dIa)	Sent POST request contains • Code = 2(POST) • Uri-Path: <sclBase> • Uri-Path: applications • Uri-Path: <app> • Uri-Path: subscriptions • Payload: subscription resource <sub> to be created • Content Format option = 41 (application/xml)
	3	Check (dIa)	SCL sends response containing: • Code = 65(2.01 Created) • Location-Path: <sclBase> • Location-Path: applications • Location-Path: <app>

Interoperability Test Description			
			• Location-Path: subscriptions • Location-Path: <sub> • The same Message ID as that of the previous request • Content Format option = 41 (application/xml) • Payload: subscriptionCreateResponse representation
	4	Verify (dIa)	M2M DA indicates successful operation

7.5.5 SubscriptionNotifyRequest

Interoperability Test Description			
Identifier:	TD_M2M_COAP_05		
Objective:	M2M GSCL sends notification(s) via subscriptionNotifyRequest (CoAP POST) and DA returns subscriptionNotifyResponse		
Configuration:	M2M_CFG_01		
References:	[5] 10.25.7, Annex D [**Erreur ! Source du renvoi introuvable.**] 9.3.2.8.19		
Pre-test conditions:	• DA has created an Application resource <app> on SCL • DA has created subscription <sub> to <app> on SCL		
Test Sequence:	**Step**	**Type**	**Description**
GG2@53	1	Stimulus	M2M DA is requested to send a applicationUpdateRequest (CoAP PUT)
	2	Check (dIa)	Sent PUT request contains • Code = 3 (PUT) • Uri-Path: <sclBase> • Uri-Path: applications • Uri-Path: <app> • Payload: modified application resource (e.g. modifies aPoc attribute) • Content Format option = 41 (application/xml)
	3	Check (dIa)	Server sends response containing: • Code = 68 (2.04 Changed) • The same Message ID as that of the previous request • Content Format option = 41 (application/xml) • Payload: applicationUpdateResponse representation
	4	Verify (dIa)	M2M DA indicates successful operation
	5	Verify (dla)	SCL sends subscriptionNotifyRequest (CoAP POST)
	6	Check (dla)	Sent POST request contains • Type = 0 (CON) • Code = 2(POST) • Uri-Path: contact attribute of <sub> • Payload: notify structure for <app> • Content Format option = 41 (application/xml)
	7	Verify (dla)	M2M DA sends subscriptionNotifyResponse
	8	Check (dla)	M2M DA sends response containing: • Code = 65(2.01 Created) • The same Message ID as that of the previous request • Content Format option = 41 (application/xml) • Payload: subscriptionNotifyResponse representation
	9	Verify (dIa)	M2M DA indicates updated value for <app>

7.5.6 SubscriptionDeleteRequest

Interoperability Test Description			
Identifier:	TD_M2M_COAP_06		
Objective:	M2M DA cancels subscription via an subscriptionDeleteRequest (CoAP DELETE)		
Configuration:	M2M_CFG_01		
References:	[5] 10.25.5, Annex D [**Erreur ! Source du renvoi introuvable.**] 9.3.2.8.19		
Pre-test conditions:	• DA has created an Application resource <app> on SCL • DA has created subscription <sub> to <app> on SCL		
Test Sequence:	**Step**	**Type**	**Description**
	1	Stimulus	M2M DA is requested to send a subscriptionDeleteRequest (CoAP DELETE)
	2	Check (dIa)	Sent DELETE request contains • Code = 4 (DELETE) • Uri-Path: <sclBase> • Uri-Path: applications • Uri-Path: <app> • Uri-Path: subscriptions • Uri-Path: <sub>
	3	Check (dIa)	SCL sends response containing: • Code = 66(2.02 Deleted) • The same Message ID as that of the previous request
	4	Verify (dIa)	M2M DA indicates successful operation

7.5.7 ApplicationDeleteRequest

Interoperability Test Description			
Identifier:	TD_M2M_COAP_07		
Objective:	M2M DA de-registers by deleting application resource via an applicationDeleteRequest (CoAP DELETE)		
Configuration:	M2M_CFG_01		
References:	[5] 10.8.5, Annex D [**Erreur ! Source du renvoi introuvable.**] 9.3.2.8		
Pre-test conditions:	• DA has created an Application resource <app> on SCL		
Test Sequence:	**Step**	**Type**	**Description**
	1	Stimulus	M2M DA is requested to send a applicationDeleteRequest (CoAP DELETE)
	2	Check (dIa)	Sent DELETE request contains • Code = 4 (DELETE) • Uri-Path: <sclBase> • Uri-Path: applications • Uri-Path: <app>
	3	Check (dIa)	Server sends response containing: • Code = 66 (2.02 Deleted) • The same Message ID as that of the previous request
	4	Verify (dIa)	M2M DA indicates successful operation

7.5.8 TargetID containing several path segments

Interoperability Test Description			
Identifier:	TD_M2M_COAP_08		
Objective:	Handle contentInstanceRetrieveRequest with targetID containing several path segments		
Configuration:	M2M_CFG_01		
References:	[5] 10.19.3, Annex D [**Erreur ! Source du renvoi introuvable.**] 9.3.2.15		
Pre-test conditions:	• DA has created an Application resource <app> on SCL • DA has created container <container1> on SCL via containerCreateRequest • DA has created a contentInstance resource **`/<container1>/contentInstances/<test>`** on SCL via contentInstanceCreateRequest		
Test Sequence:	**Step**	**Type**	**Description**
	1	Stimulus	M2M DA is requested to send a contentInstanceRetrieveRequest (CoAP GET) on resource **`<container1>/contentInstances/<test>`**
	2	Check (dIa)	Sent GET request contains • Code = 1(GET) • Uri-Path: <sclBase> • Uri-Path: applications • Uri-Path: <app> • Uri-Path: containers • Uri-Path: <container1> • Uri-Path: contentInstances • Uri-Path: <test> •
	3	Check (dIa)	SCL sends response containing: • Code = 69(2.05 Content) • The same Message ID as that of the previous request • Content Format option = 41 (application/xml) • Payload: contentInstanceRetrieveResponse representation
	4	Verify (dIa)	M2M DA indicates successful operation

7.5.9 TargetID containing several query options

Interoperability Test Description			
Identifier:	TD_M2M_COAP_09		
Objective:	Handle contentInstanceRetrieveRequest with targetID containing several query options		
Configuration:	M2M_CFG_01		
References:	[5] 10.19.3, Annex D [**Erreur ! Source du renvoi introuvable.**] 9.3.2.15		
Pre-test conditions:	• DA has created an Application resource <app> on SCL • DA has created a collection of resources <collec> with filter criterias (criteria1, criteria2) on SCL using contentInstancesCreateRequest • DA has created several resources in this collection via contentInstantCreateRequest		
Test Sequence:	**Step**	**Type**	**Description**
	1	Stimulus	M2M DA is requested to send a contentInstancesRetrieveRequest (CoAP GET) on resource <**collec**> with filter criterias (criteria1, criteria2)
	2	Check (dIa)	Sent GET request contains • Code = 1(GET) • Uri-Path: <sclBase> • Uri-Path: applications • Uri-Path: <app> • Uri-Path: <collec>

Interoperability Test Description			
			• Uri-Query: criteria1, value1 • Uri-Query: criteria2, value2 • Content Format option
	3	Check (dIa)	SCL sends response containing: • Code = 69(2.05 Content) • The same Message ID as that of the previous request
	4	Verify (dIa)	M2M DA indicates successful operation

7.5.10 TargetID using partial addressing

Interoperability Test Description			
Identifier:	TD_M2M_COAP_10		
Objective:	Handle contentInstanceRetrieveRequest with targetID using partial addressing to fetch a contentInstance attribute		
Configuration:	M2M_CFG_01		
References:	[5] 10.19.3, Annex D [**Erreur ! Source du renvoi introuvable.**] 9.3.2.15		
Pre-test conditions:	• DA has created an Application resource <app> on SCL • DA has created container <container1> on SCL via containerCreateRequest • DA has created a **/<container1>/contentInstances/<test>** resource on SCL via contentInstanceCreateRequest • DA performs a partial addressing request to fetch the M2M 'content' attribute of the contentInstance resource		
Test Sequence:	**Step**	**Type**	**Description**
	1	Stimulus	M2M DA is requested to send a contentInstanceRetrieveRequest (CoAP GET) to the M2M *content* attribute within the contentInstance resource <**test>**
	2	Check (dIa)	Sent GET request contains • Code = 1(GET) • Uri-Path: <sclBase> • Uri-Path: applications • Uri-Path: <app> • Uri-Path containers • Uri-Path: <container1> • Uri-Path contentInstances • Uri-Path: <test> • Uri-Path: content •
	3	Check (dIa)	SCL sends response containing: • Code = 69(2.05 Content) • The same Message ID as that of the previous request • Content Format option = 41 (application\xml) • Payload: contentInstanceRetrieveResponse representation containing content attribute
	4	Verify (dIa)	M2M DA indicates successful operation

7.5.11 Announcement

Interoperability Test Description			
Identifier:	TD_M2M_COAP_11		
Objective:	M2M DA registration to GSCL with GSCL Announcement to NSCL		
Configuration:	M2M_CFG_02		
References:	[5] 10.9.2, Annex D [**Erreur ! Source du renvoi introuvable.**] 9.3.2.28		
Pre-test conditions:	• GSCL has registered to NSCL as <gscl>		
Test Sequence:	**Step**	**Type**	**Description**

Interoperability Test Description			
	1	Stimulus	M2M DA is requested to send a applicationCreateRequest (CoAP POST) with AnnounceTo option activated
	2	Check (dIa)	Sent POST request contains • Code = 2(POST) • Uri-Path: <gsclBase> • Uri-Path: applications • Payload: application resource <app_ann> to be created • Content Format option = 41 (application\xml)
	3	Check (dIa)	GSCL sends response containing: • Code = 65(2.01 Created) • Location-Path: <gsclBase> • Location-Path: applications • Location-Path: <app_ann> • The same Message ID as that of the previous request • Content Format option = 41 (application\xml) • Payload: applicationCreateResponse representation
	4	Verify (dIa)	M2M DA indicates successful operation
	5	Verify (mId)	M2M GSCL sends applicationAnncCreateRequest (CoAP POST) to M2M NSCL
	6	Check (mId)	Sent POST request contains • Code = 2(POST) • Uri-Path: <nsclBase> • Uri-Path: scls • Uri-Path: <gscl> • Uri-Path: applications • Uri-Path: <app_ann>Annc • Payload: applicationAnnc resource <app_ann>Annc to be created • Content Format option = 41 (application\xml)
	7	Check (mId)	NSCL sends response containing: • Code = 65(2.01 Created) • Location-Path: <nsclBase> • Location-Path: scls • Location-Path: <gscl> • Location-Path: applications • Location-Path: <app_ann>Annc • The same Message ID as that of the previous request
	8	Verify (mId)	NSCL indicates announced resource <app_ann>Annc

7.5.12 Multihop retrieval using Proxy-Uri and aPoC

Interoperability Test Description			
Identifier:	TD_M2M_COAP_12		
Objective:	M2M NA multi-hop resource retrieval using Proxy-URI (CoAP proxy)		
Configuration:	M2M_CFG_02		
References:	[5] 10.19.3, Annex D 1.5 [**Erreur ! Source du renvoi introuvable.**] 9.3.2.15		
Pre-test conditions:	• DA has created an announceable Application resource <app_ann> on GSCL which has aPoC attribute configured to enable GSCL to DA re-targeting • GSCL has announced <app_ann> to NSCL • DA offers the resource `/test` • NA has discovered the resource `/test` offered by DA		
Test Sequence:	**Step**	**Type**	**Description**
	1	Stimulus	M2M NA is requested to send a (CoAP GET) to NSCL for resource `/test` on DA leveraging GSCL aPoC re-targeting capability
	2	Check (mIa)	Sent GET request contains • Code = 1(GET) • Proxy-Uri:

Interoperability Test Description			
			coap://<gsclBase>/applications/<app_ann>/test •
	3	Verify (mId)	NSCL CoAP proxies the request to GSCL
	4	Check (mId)	CoAP Proxied GET request contains • Code = 1(GET) • Uri-Path: <gsclBase> • Uri-Path: applications • Uri-Path: <app_ann> • Uri-Path: test •
	5	Verify (dIa)	GSCL aPoC proxies (i.e. re-targets) request to DA /test resource
	6	Check (dIa)	aPoC Proxied GET request contains • Code = 1(GET) • Uri-Path: test
	7	Check (dIa)	DA sends response containing: • Code = 69(2.05 Content) • The same Message ID as that of the previous request • Content Format option = 41 (application\xml) • Payload: content of resource `/test`
	8	Check (mId)	GSCL aPoC proxies response to NSCL
	9	Verify (mIa)	NSCL CoAP proxies the response to NA
	10	Check (mIa)	Proxied response contains: • Code = 69(2.05 Content) • The same Message ID as that of the previous request • Content Format option = 41 (application\xml) • Payload: content of resource `/test`
	11	Verify (mIa)	M2M NA indicates successful operation

7.5.13 Multihop retrieval using m2mPocs

Interoperability Test Description			
Identifier:	TD_M2M_COAP_12		
Objective:	M2M NA multi-hop resource retrieval using m2mPocs (M2M proxy)		
Configuration:	M2M_CFG_02		
References:	[5] 10.19.3, Annex D [**Erreur ! Source du renvoi introuvable.**] 7.3, 9.2.1.9, 9.2.3.4, 9.2.3.24, 9.2.3.25, 9.3.2.21		
Pre-test conditions:	• GSCL has created an m2mPoc <test_poc> on NSCL to enable NSCL to M2M proxy requests from NA to GSCL • DA has created an Application resource <app > on GSCL • DA has created a contentInstance resource <test> on GSCL		
Test Sequence:	**Step**	**Type**	**Description**
	1	Stimulus	M2M NA is requested to send a contentInstanceRetrieveRequest (CoAP GET) to NSCL for resource `<test>`
	2	Check (mIa)	Sent GET request contains • Code = 1(GET) • Uri-Path: <gsclBase> • Uri-Path: applications • Uri-Path: <app> • Uri-Path: containers • Uri-Path: <container1> • Uri-Path: contentInstances • Uri-Path: <test> •
	3	Verify (mId)	NSCL M2M proxies the request to GSCL using m2mpoc

Interoperability Test Description			
			information
	4	Check (mId)	Proxied GET request contains • Code = 1(GET) • Uri-Path: <gsclBase> • Uri-Path: applications • Uri-Path: <app> • Uri-Path: containers • Uri-Path: <container1> • Uri-Path: contentInstances • Uri-Path: <test> •
	5	Check (mId)	GSCL sends contentInstanceRetrieveResponse containing: • Code = 69(2.05 Content) • The same Message ID as that of the previous request • Content Format option = 41 (application\xml) • Payload: content of resource `<test>`
	6	Verify (mIa)	NSCL M2M proxies the response to NA
	7	Check (mIa)	Proxied contentInstanceRetrieveResponse contains: • Code = 69(2.05 Content) • The same Message ID as that of the previous request • Content Format option = 41 (application\xml) • Payload: content of resource `<test>`
	8	Verify (mIa)	M2M NA indicates successful operation

Change History

Document history		
0.0.1	21.09.2012	First Draft
0.0.2	01.10.2012	Formal changes in section 7.2 "Core Link Format" Updated section 7.4 "Observing Resources": • Used empty observe option instead of zero (see tests 01-2, 03-4) • Added note about timing (see tests 03-4) • Added check for retransmission updates (see tests 04-2) • Formal changes
	04.10.2012	12 Test description added for CoAP CORE & Reverse proxy Test set up for reverse proxy added
0.0.3	05.10.2012	Added tests TD_COAP_LINK_03,04,05,06,07,08,09,10 Added tests TD_COAP_OBS_06 and TD_COAP_OBS_07 Updated section 4.5 "Test summary – Optional CoAP tests" Updated table 6 "Resource offered by CoAP Servers" 12 Test description reviewed & 7 test description modified based on review
	09.10.2012	6 Test description modified based on review made on 05.10.2012 Clarification and updates in Block-wise
		Clarification and updates in Block-wise
0.0.4	12.10.2012	Fixed TD_COAP_OBS_06 and TD_COAP_OBS_07 1 test description added in CORE and modification made from the comments received during the conf call of 11/10/2012 Fixed ETSI M2M section for all TD handling with Location-Path in response other than POST
0.0.5	19.10.2012	Test architecture for reverse proxy modified. TD_COAP_CORE_01 to TD_COAP_CORE_29 were rechecked and objectives were harmonised. Link format reviewed Observe reviewed and modifaction done for all 7 TD Preamble added for all 7 TD in observe
0.0.6	23.10.2012	Fixed link format and observe test.
0.0.7	24.10.2012	Editorial changes.
	07.11.2012	Changes made in TD_COAP_CORE_09 "Content -type" is replaced with"content format" according to coap-12
0.0.8	09.11.2012	Changes made in TD_COAP_OBS tests to adopt CON type as a default type for registrations and notifications messages Added TD_COAP_OBS_02 for NON messages testing with observe Renumbered TD_COAP_OBS tests
0.0.9	15.11.2012	Updated M2M xml resource representations Corrections/clarifications to the M2M tests in section 7.5. Fixed TD_COAP_LINK_03 step 2: rt="" replaced by rt="*" Added TD_COAP_OBS_09: "Update of the observed resource" Fixed TD_COAP_OBS tests: CON requests were not ACKed. New step added and step numbers updated accordingly. Fixed TD_COAP_OBS tests: message type values were wrong. Added further information about message content in TD_COAP_OBS_07 and TD_COAP_OBS_08 Moved the CORE TDs 24 to 29 (related to Reverse Proxy) in the Optional table.
0.0.9a	07.11.2012	Updates to Table for the M2M XML representations Correction to TD_M2M_COAP_08
0.0.9d	22.11.2012	TD_COAP_CORE_09: removed the Token option (overlap with TD_COAP_CORE_11) TD_COAP_CORE_21: added checks on the absence of the ETag option in step 2 and on the presence of the payload in steps 3, 7 and 12 TD_COAP_OBS_{01,03,04,05,06,07}: clarified the stimulus is a confirmable request TD_COAP_OBS_03: rewording of step 7 TD_COAP_OBS_04: clarified that step 10 is not a stimulus TD_COAP_OBS_05: clarified that the server should keep retransmitting until Max-Age is elapsed TD_COAP_OBS_{07,08,09}: clarified that deletion (or update) of the observed resource should be made from an external source (so that the client is not aware of the deleteon/update) TD_COAP_OBS_{06,07,08}: added a Verifi step at the end of the test to ensure that the client does not display any further notifications

0.0.9e	23.11.2012	Order of tests were changed between TD_COAP_CORE_01 to 08 to ease automation : DELETE transcation was made available before PUT & POST trascation Test summary was updated with new order Resource table updated TD_COAP_CORE_01, 05, 10, 12, 13, 14, 15 & 27: pre-test conditions updated with 'resource content is not empty' TD_COAP_CORE_09, 10, 11, 12, 13 , 15, 16, , 17, 21, 22, 27, 29: 'Payload = content of requested resource' is replaced with 'Not empty payload' TD_COAP_CORE_20 & 26 : content-format value changed from 1 to 0 TD_COAP_CORE_21, 27 & 29 : resource /test changed to /validate TD_COAP_CORE_23, 24 & 28 : resource /test changed to /create1, /create2 & /create3 respectively
0.0.10	26.11.2012	Fixed resource name and test objective in TD_COAP_CORE_23
0.0.11	26.11.2012	Editorial changes + remove the test TD_COAP_LINK_10
0.0.12	26.11.2012	Fixed some typos
0.0.13	29.11.2012	TD_COAP_CORE_22: added an intermediate GET request to retrieve the new ETag of the updated resource between the two PUT requests (the coap draft does not require the ETag option to be present in 2.04 responses) TD_COAP_LINK_05: replaced the pre-condition <if=""> with <no if attribute> (the link-format BNF does not allow empty if attribute) Resources offered by servers under test : - clarified that the server sends confirmable notifications for the observable resource /obs - added a new observable resource: /obs-non which produces non-confirmable notifications TD_COAP_OBS_02: use the /obs-non resource instead of the /obs (so as to produce non-confirmable notifications) TD_COAP_OBS_{01,03,04,05,06,07,08,09}: clarified that the server is configured to send confirmable notifications for the /obs resource