IoT CoAP Plugtests;
Paris, France;
24 - 25 March 2012

World Class Standards

ETSI

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

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

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

Important notice

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

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

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

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

Copyright Notification

Contents

ProbeIT – ETSI declaration

The FP7 Probe-IT project[1] (hereinafter: "ProbeIT") carries out comprehensive assessments of IoT systems and related interoperability testing methodologies used in order to verify their benefits and to pave the way for market implementation.

The ETSI Centre for Testing and Interoperability (hereinafter "ETSI CTI") provides direct support and assistance to ETSI technical committees on the application of validation and testing techniques in standards.

ETSI CTI is cooperating with the ProbeIT in order to facilitate IoT interoperability event(s) and other testing activities. ETSI CTI and ProbeIT have jointly contributed to the development of this document.

---

[1] FP7 Probe-IT (Pursuing Roadmap and Benchmark in Internet of things). http://www.probe-it.eu. This is an FP7 project funded by the European Union

1   Scope

This document forms the guidelines to lead the technical organization of the 1st IoT CoAP Plugtests event, in Paris, from 24 to 25 March 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, which are describing the scenarios, which the participants will follow to perform the 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-08

[2]   CoRE Link Format; draft-ietf-core-link-format-11

[3]   Observing Resources in CoAP; draft-ietf-core-observe-04

[4]   Blockwise transfers in CoAP; draft-ietf-core-block-08

3   Abbreviations

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

IoT   Internet of Things

Probe-IT   Pursuing Roadmap and Benchmark in IoT

RST   Reset

CON   Confirmable

NON   Non-Confirmable

ACK   Acknowledgement

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, one meesage may be checked during a 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 Interoperablity 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 receipt of protocol messages on reference points, with valid content. This "check" event type corresponds to the interoperability testing with conformance check method

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

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

4.2   Tooling

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

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

- Except for the "check" events, the verification of the message conformity 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
<gr> = group	CORE	Core protocol
	LINK	CoRE Link Format
	BLOCK	Blockwise transfers
	OBS	Observing Ressources
<nn> = sequential number		01 to 99

4.4   Test Summary – Mandatory Tests

Table 2: Mandatory Tests

1	TD_COAP_CORE_01	Perform GET transaction (CON mode)
2	TD_COAP_CORE_02	Perform POST transaction (CON mode)
3	TD_COAP_CORE_03	Perform PUT transaction (CON mode)
4	TD_COAP_CORE_04	Perform DELETE transaction (CON mode)
5	TD_COAP_CORE_05	Perform GET transaction (NON mode)
6	TD_COAP_CORE_06	Perform POST transaction (NON mode)
7	TD_COAP_CORE_07	Perform PUT transaction (NON mode)
8	TD_COAP_CORE_08	Perform DELETE transaction (NON mode)
9	TD_COAP_CORE_09	Perform GET transaction with delayed response (CON mode, no piggyback)
10	TD_COAP_CORE_10	Handle request containing Token option
11	TD_COAP_CORE_11	Handle request not containing Token option
12	TD_COAP_CORE_12	Handle request containing several Uri-Path options
13	TD_COAP_CORE_13	Handle request containing several Uri-Query options
14	TD_COAP_CORE_14	Interoperate in lossy context (CON mode, piggybacked response)
15	TD_COAP_CORE_15	Interoperate in lossy context (CON mode, delayed response)
16	TD_COAP_CORE_16	Perform GET transaction with delayed response (NON mode)

4.5   Test Summary – Optional 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_BLOCK_01	Handle GET blockwise transfer for large resource (early negotiation)
4	TD_COAP_BLOCK_02	Handle GET blockwise transfer for large resource (late negotiation)
5	TD_COAP_BLOCK_03	Handle PUT blockwise transfer for large resource
6	TD_COAP_BLOCK_04	Handle POST blockwise transfer for large resource
7	TD_COAP_OBS_01	Handle resource observation
8	TD_COAP_OBS_02	Stop resource observation
9	TD_COAP_OBS_03	Client detection of deregistration (Max-Age)
10	TD_COAP_OBS_04	Server detection of deregistration (client OFF)
11	TD_COAP_OBS_05	Server detection of deregistration (explicit RST)

5   Basic Configuration

5.1   Resources offered by servers under test

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

Table 4: Resources offered by 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
/seg1/seg2/seg3	Long path ressource	TD_COAP_CORE_12
/query	Ressource accepting query parameters	TD_COAP_CORE_13
/separate	Ressource 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	TD_COAP_OBS_01 TD_COAP_OBS_02 TD_COAP_OBS_03 TD_COAP_OBS_04 TD_COAP_OBS_05
/.well-known/core	CoRE Link Format	TD_COAP_LINK_01 TD_COAP_LINK_02

Note on resource sizes:

- Ressources 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

6   Test Configurations

This section defines the different test configurations.

6.1   Test Configuration 1 (CoAP_CFG_01)
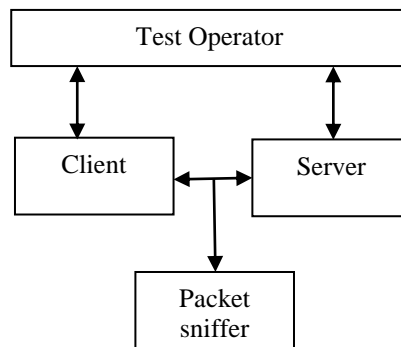
Figure 1: Basic Face 2 Face Configuration

6.2   Test Configuration 2 (CoAP_CFG_02)
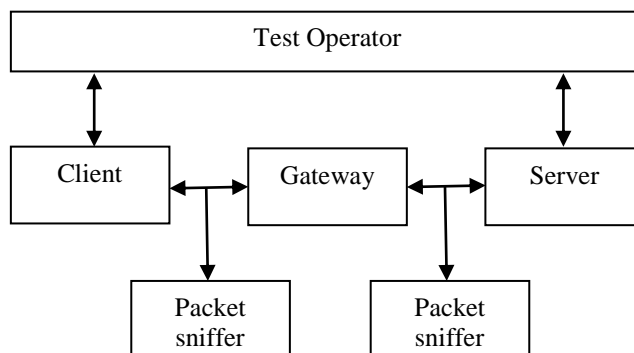
Figure 2: Basic Face 2 Face Configuration in lossy context

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

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 explicitely stated in the test description, but implementation should be prepared to handle it

- Use of Token shall be avoided except if explicitely 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] 4.4.1, 4.4.3, 5.8.1		
Pre-test conditions:	• Server offers the resource **/test** 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	Sent request contains Type value indicating 0 and Code value indicating 1
	3	check	Server sends response containing: • Code = 69(2.05 Content) • The same Message ID as that of the previous request • Content type option
	4	verify	Client displays the received information

Interoperability Test Description			
Identifier:	TD_COAP_CORE_02		
Objective:	Perform POST transaction (CON mode)		
Configuration:	CoAP_CFG_01		
References:	[1] 4.4.1, 4.4.3, 5.8.2		
Pre-test conditions:	• Server accepts creation of new resource on **/test** (resource does not exists 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 type option
	2	check	Sent request contains Type value indicating 0 and Code value indicating 2
	3	verify	Server displays received information
	4	check	Server sends response containing: • Code = 65(2.01 Created) • The same Message ID as that of the previous request
	5	verify	Client displays the received response

Interoperability Test Description			
Identifier:	TD_COAP_CORE_03		
Objective:	Perform PUT transaction (CON mode)		
Configuration:	CoAP_CFG_01		
References:	[1] 4.4.1, 4.4.3, 5.8.3		
Pre-test conditions:	• Server offers a resource **/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 type option
	2	check	Sent request contains Type value indicating 0 and Code value indicating 3
	3	verify	Server displays received information
	4	check	Server sends response containing: • Code = 68(2.04 Changed) • The same Message ID as that of the previous request
	5	verify	Client displays the received response

Interoperability Test Description			
Identifier:	TD_COAP_CORE_04		
Objective:	Perform DELETE transaction (CON mode)		
Configuration:	CoAP_CFG_01		
References:	[1] 4.4.1, 4.4.3, 5.8.4		
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	Sent request contains Type value indicating 0 and Code value indicating 4
	3	check	Server sends response containing: • Code = 66(2.02 Deleted) • The same Message ID as that of the previous request
	4	verify	Client displays the received information

Interoperability Test Description			
Identifier:	TD_COAP_CORE_05		
Objective:	Perform GET transaction (NON mode)		
Configuration:	CoAP_CFG_01		
References:	[1] 4.4.2, 5.8.1		
Pre-test conditions:	• Server offers a **/test** resource 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	Sent request contains Type value indicating 1 and Code value indicating 1
	3	check	Server sends response containing: • Type = 1(NON) • Code= 69(2.05 Content) • Content type option
	4	verify	Client displays the received information

Interoperability Test Description			
Identifier:	TD_COAP_CORE_06		
Objective:	Perform POST transaction (NON mode)		
Configuration:	CoAP_CFG_01		
References:	[1] 4.4.2, 5.8.2		
Pre-test conditions:	• Server accepts creation of new resource on **/test** (resource does not exists 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 type option
	2	check	Sent request contains Type value indicating 1 and Code value indicating 2
	3	verify	Server displays the received information
	4	check	Server sends response containing: • Type = 1(NON) • Code = 65(2.01 Created)
	5	verify	Client displays the received response

Interoperability Test Description			
Identifier:	TD_COAP_CORE_07		
Objective:	Perform PUT transaction (NON mode)		
Configuration:	CoAP_CFG_01		
References:	[1] 4.4.2, 5.8.3		
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 type option
	2	check	Sent request contains Type value indicating 1 and Code value indicating 3
	3	verify	Server displays the received information
	4	check	Server sends response containing: • Type = 1(NON) • Code = 68(2.04 Changed)
	5	verify	Client displays the received response

Interoperability Test Description			
Identifier:	TD_COAP_CORE_08		
Objective:	Perform DELETE transaction (NON mode)		
Configuration:	CoAP_CFG_01		
References:	[1] 4.4.2, 5.8.4		
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) • Code = 4(DELETE)
	2	check	Sent request contains Type value indicating 1 and Code value indicating 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_09		
Objective:	Perform GET transaction with a separate response		
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 and which cannot be acknowledged in a piggy-backed way.		
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 (CON) • Code = 1 (GET) • Client generated Message ID
	3	Check	Server sends response containing: • Type = 2 (ACK) • message ID same as the request • empty Payload
	4	Check	Server sends response containing: • Type = 0 (CON) • Code = 69 (2.05 content) • Payload = Content of the requested resource • Content type option
	5	Check	Client sends response containing: • Type = 2 (ACK) • message ID same as the response • empty Payload
	6	Verify	Client displays the response

Interoperability Test Description			
Identifier:	TD_COAP_CORE_10		
Objective:	Handle request containing Token option		
Configuration:	CoAP_CFG_01		
References:	[1] clause 2.2 ,5.8.1, 5.10.1		
Pre-test conditions:	• Server offers a **/test** resource 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	Sent request must contain: • Type = 0 (CON) • Code = 1 (GET) • Client generated Token value • Length of the token should be between 1 to 8 B • Option Type = Token
	3	Check	Server sends response containing: • Code = 69 (2.05 content) • Length of the token should be between 1 to 8 B • Token value same as the requested • Payload = Content of the requested resource • Content type option
	4	Verify	Client displays the response

Interoperability Test Description			
Identifier:	TD_COAP_CORE_11		
Objective:	Handle request not containing Token option		
Configuration:	CoAP_CFG_01		
References:	[1] clause 2.2 ,5.8.1, 5.10.1		
Pre-test conditions:	• Server offers a **/test** resource that handles GET		
Test Sequence:	**Step**	**Type**	**Description**
	1	stimulus	Client is requested to send a confirmable GET request to server's resource not containg Token option
	2	Check	Sent request must contain: • Type = 0 (CON) • Code = 1 (GET) • No Token option
	3	Check	Server sends response containing: • Code = 69 (2.05 content) • No Token option • Payload = Content of the requested resource • Content type option
	4	Verify	Client displays the response

Interoperability Test Description			
Identifier:	TD_COAP_CORE_12		
Objective:	Handle request containing several URI-Path options		
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		
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 (CON) • Code = 1 (GET) • Option type = URI-Path (one for each path segment)
	3	Check	Server sends response containing: • Code = 69 (2.05 content) • Payload = Content of the requested resource • Content type option
	4	Verify	Client displays the response

Interoperability Test Description			
Identifier:	TD_COAP_CORE_13		
Objective:	Handle request containing several URI-Query options		
Configuration:	CoAP_CFG_01		
References:	[1] clause 5.4.5, 5.10.2,6.5		
Pre-test conditions:	• Server offers a **/query** resource		
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	Sent request must contain: • Type = 0 (CON) • Code = 1 (GET) • Option type = URI-Query (More than one query parameter)
	3	Check	Server sends response containing: • Type = 0/2 (CON/ACK) • Code = 69 (2.05 content) • Payload = Content of the requested resource • Content type option
	4	Verify	Client displays the response

Interoperability Test Description			
Identifier:	TD_COAP_CORE_14		
Objective:	Interoperate in lossy context (CON mode, piggybacked response)		
Configuration:	CoAP_CFG_02		
References:	[1] clause 4.4.1, 5.2.1		
Pre-test conditions:	• Gateway is introduced and configured to produce packet loss • Server offers a **/test** resource 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) • Payload = Content of the requested resource • Content type 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

Interoperability Test Description			
Identifier:	TD_COAP_CORE_15		
Objective:	Interoperate in lossy context (CON mode, delayed response)		
Configuration:	CoAP_CFG_02		
References:	[1] clause 4.4.1, 5.2.1		
Pre-test conditions:	• Gateway is introduced and configured to produce packet loss • Server offers a **/separate** resource which cannot be served immediately and which cannot be acknowledged in a piggy-backed way.		
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) • message ID same as the request • empty Payload
	4	Check	Server sends response containing: • Type = 0 (CON) • Code = 69 (2.05 content) • Payload = Content of the requested resource • Content type option
	5	Check	Client sends response containing: • Type = 2 (ACK) • message ID same as the response • 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

Interoperability Test Description			
Identifier:	TD_COAP_CORE_16		
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 confirmable GET request to server's resource
	2	Check	Sent request must contain: • Type = 1 (NON) • Code = 1 (GET) • Client generated Message ID
	3	Check	Server does not send response containing: • Type = 2 (ACK) • message ID same as the request • empty Payload
	4	Check	Server sends response containing: • Type = 1 (NON) • Code = 69 (2.05 content) • Payload = Content of the requested resource • Content type option
	5	Verify	Client displays the response

7.2 CoRE Link Format

Identifier:	TD_COAP_LINK_01		
Objective:	Access to well-known interface for resource discovery		
Configuration:	CoAP_CFG_01		
References:	[2]		
Pre-test conditions:	• Client 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	chec	Client sends a GET request to Server for /.well-known/core resource
	3	check	Server sends response containing: Content-Type option indicating 40 (application/link-format) payload indicating all the links available on Server
	4	verify	Client displays the list of resources available on Server

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-Type 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			

## 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 containing Block2 option indicating block number 0 and desired block size
	3	check	Server sends response containing Block2 option indicating block number and size
	4	check	Client send GET requests for further blocks
	5	check	Each request contains Block2 option indicating block number of the next block and size of the last received block
	6	check	Server sends further responses containing Block2 option indicating block number and size
	7	verify	Client displays the received information

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 response containing Block2 option indicating block number and size
	4	check	Client send GET requests for further blocks
	5	check	Each request contains Block2 option indicating block number of the next block and size of the last received block or the desired size of next block
	6	check	Server sends further responses containing Block2 option indicating block number and size
	7	verify	Client displays the received information

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 block number 0 and block size
	3	check	Client sends further requests containing Block1 option indicating block number and size
	4	verify	Server indicates presence of the complete updated resource /large-update

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 on Server
	2	check	Client sends a POST request containing Block1 option indicating block number 0 and block size
	3	check	Client sends further requests containing Block1 option indicating block number and size
	4	verify	Server indicates presence of the complete new resource

7.4   Observing Resources

Interoperability Test Description			
Identifier:	TD_COAP_OBS_01		
Objective:	Handle resource observation		
Configuration:	CoAP_CFG_01		
References:	[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)		
Test Sequence:	**Step**	**Type**	**Description**
	1	stimulus	Client is requested to observe resource /obs on Server
	2	check	Client sends a GET request containing Observe option indicating 0
	3	check	Server sends response containing Observe option
	4	verify	Client displays the received information
	5	check	Server sends response containing Observe option indicating increasing values, as resource changes
	6	verify	Client displays the updated information

Interoperability Test Description			
Identifier:	TD_COAP_OBS_02		
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) • Client is observing /obs on Server		
Test Sequence:	**Step**	**Type**	**Description**
	1	stimulus	Client is requested to stop observing resource /obs on Server
	2	check	Client sends GET request not containing Observe option
	3	check	Server sends response not containing Observe option
	4	verify	Client displays the received information
	5	check	Server does not send further response
	6	verify	Client does not display updated information

Interoperability Test Description			
Identifier:	TD_COAP_OBS_03		
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) • Client is observing /obs on Server		
Test Sequence:	**Step**	**Type**	**Description**
	1	stimulus	Server is rebooted
	2	check	Server does not send notifications
	3	verify	Client does not display updated information
	4	verify	After Max-Age expiration, Client sends a new GET with Observe option for Server's observable resource
	5	check	Sent request contains Observe option indicating 0
	6	check	Server sends response containing Observe option
	7	verify	Client displays the received information
	8	check	Server sends response containing Observe option indicating increasing values, as resource changes
	9	verify	Client displays the updated information

Interoperability Test Description			
Identifier:	TD_COAP_OBS_04		
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) • Client is observing /obs on Server		
Test Sequence:	**Step**	**Type**	**Description**
	1	stimulus	Client is switched off
	2	check	Server's confirmable responses are not acknowledged
	3	verify	After some delay, Server does not send further responses

Identifier:	TD_COAP_OBS_05		
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) • Client is observing /obs on Server		
Test Sequence:	**Step**	**Type**	**Description**
	1	stimulus	Client is rebooted
	2	check	Server sends response containing Observe option
	3	verify	Client discards response and does not display information
	4	check	Client sends RST to Server
	5	check	Server does not send further response

Change History

		Document history
0.0.1	05.01.2012	First Draft
0.0.2	09.01.2012	First sample Test Description added
0.0.3	10.01.2012	Test objectives added
0.0.4	18.01.2012	[BUPT] 8 Test Descriptions added
0.0.5	20.01.2012	[BUPT] TPLan notation deleted; Several mistakes in the test sequence part corrected
0.0.6	18.01.2012	[IRISA] 7 Test Descriptions added
0.0.7	20.01.2012	[IRISA] Internally reviewed and Test Descriptions updated
0.0.8	26.01.2012	[IRISA] A figure added in Test bed architecture
0.0.9	18.01.2012	[ETSI] Added Test Descriptions for Link Format Added Test Descriptions for Blockwise Transfer Added Test Descriptions for Observe
0.0.10	27.01.2012	Merged various versions
0.0.11	30.01.2012	Merged some steps Common IUT setup List and name server resources
0.0.11 Updated	31.01.2012	Test configuration figures updated
0.0.12	03.02.2012	Merged comments from Zach
0.0.13	28.02.2012	Fixed Content-Type value in TD_COAP_LINK_01 and TD_COAP_LINK_02 (41 -> 40) Clarified pre-conditions of TD_COAP_CORE_02 and TD_COAP_CORE_06 [IRISA] Added description of the Gateway in "lossy context" configuration Updated ProbeIT – ETSI declaration
0.0.14	01.03.2012	Refined ProbeIT description Added ACK definition Updated Block and Observe reference specs TD_COAP_CORE_05..._08: removed "different Message-ID" statements TD_COAP_LINK_02: Added note to clarify resource types values Added checks for content-type option Clarification on the use of Etag and Token options
0.0.15	08.03.2012	Added recommendations concerning payload lengths Added test TD_COAP_CORE_16
0.0.16	10.03.2012	Informative references deleted
		Empty clauses in section 5 deleted TD_COAP_CORE_16
		CON/IOP qualifiers deleted in verify/check column
	15.03.2012	TD_COAP_CORE_16 added under 5.3 for /separate
	19.03.2012	In table 4 large_update/large_create chengaed to large-update/large-create
	21.03.2012	'Need to observe' entry correct in TD_COAP_CORE_14
		'Need to observe' moved to a 'check' statement
		Further typos corrected