ETSI CTI Plugtest Report 1.0.0 (2013-01)

CoAP#2 Plugtest;
Sophia-Antipolis, France;
28 - 30 November 2012

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

1 Executive Summary

The CoAP#2 Plugtest was held from 28 to 30 November 2012 in Sophia-Antipolis.

This event was jointly organized by ETSI, IPSO Alliance and the FP7 Probe-IT project.

After the 1st CoAP Plugtest held from 24 to 25 March 2012 in Paris, France, ETSI has been requested by several participants to get another Interop event on CoAP in 2012, with an extended scope comparing to the prior event.

In addition, the ETSI Technical Committee TC M2M has published the ETSI TS 102 921 V1.1.1 including a full section dedicated to the CoAP Binding for M2M REST Resources in Annex D (normative). The TC M2M requested ETSI CTI to give an opportunity to test this aspect of the ETSI TC M2M architecture.

TC M2M started the work on TS 103 104 (Interoperability Test Specification for CoAP Binding of ETSI M2M Primitives). The document is not yet published but has been the main base for CTI to develop a Plugtest guide used for the CoAP#2 Interop event.

This event had excellent industry participation of 15 companies with 8 companies providing implementations (3 companies were there as observers) and 4 companies as part of the organising Plugtest team. Altogether there were more than 25 people at this event and 1775 interoperability tests were conducted.

The conclusions are that

- all implementations have been compatible on a basic level
- more than 97.8% of the executed tests indicated interoperability and regarding the mandatory section of the test specification 98% of possible client/server combinations were tested, which shows a very high level of maturity of the CoAP implementations
- CoAP standards are mature (This applies to the parts of base standards that were covered during the Plugtest)

2 Introduction

This plugtest aimed to test the interoperability of CoAP client and server implementations and the CoAP Binding of ETSI M2M primitives.

The implementations were connected via both IPv6 and an IPV4 test networks.

A Plugtest guide was produce containing 65 interoperability tests.

ETSI provided the interoperability tool suite of wiki, scheduling, test reporting tool and an online tool for CoAP trace validation.

The FP7 Probe-IT (represented by IRISA/Université de Rennes 1 and BUPT) provided technical expertise and a lossy gateway for testing lossy contexts.

Each day test sessions for IOP assessment were conducted. At the end of each day a wrap-up meeting was held to discuss main interoperability points of the day.

During the event the FP7 Probe-It has proposed the IoT Conformance Validation Framework to participants for live trial against their implementations.

3 Base Specifications

The following documents were used as basis for the tests:

[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".

[7] Draft-ietf-httpbis-p2-semantics-21

[8] Draft-ietf-httpbis-p4-conditional-21

4 Abbreviations

CoAP	Constrained Application Protocol
NO	Test is recorded as NOT successfully passed.
NA	Test is not applicable.
OK	Test is recorded as successfully passed.
OT	Test is recorded as not being executed due to lack of time.
Test Session	A paring of vendors that test together during a given time slot.
TSR	Test Session Report. Report created during a test session.

5 Participants

The companies which attended the Plugtest are listed in the table below.

Table 1: List of implementations that participated in the tests

#	Company
1	ETH Zurich
2	HITACHI
3	HUAWEI
4	IMINDS
5	INTECS S.P.A
6	INTERDIGITAL
7	SENSINODE
8	TZI / Uni Bremen

Table 2: List of plugtest team

#	Company	Role
1	ETSI	Organization of Plugtest, Test Network, Test Descriptions
2	IRISA/Univeristé de Rennes 1	Online Trace Validation, Test Descriptions
3	BUPT	Lossy Gateway
4	Probe-IT Project	Technical expertise

Table 3: List of Observer companies

#	Company
1	Chunghwa Telecom
2	HUFS (Korea)
3	INFRES/ Telecom Paris Tech

6 Technical and Project Management

All the information presented in this chapter is an extract of the ETSI event wiki https://services.plugtests.net/wiki/IoT-CoAP2/index.php/Main_Page (Access for registered people only).

6.1 Test Plan

The test plan containing 65 interoperability tests was developed by ETSI CTI together with Probe-IT and IPSO Alliance. The coverage of the specifications has considerably been extended regarding the former CoAP Plugtest event of March 2012 where 27 Interoperability Tests were proposed.

During the regular conference calls which were held as part of the event preparation, companies could propose additional tests. The tests were grouped in 3 categories: mandatory tests, optional tests and ETSI M2M tests. The features covered by all tests are listed below:

- CoAP Testing based on updated base specifications (updated since CoAP#1 event)
    - Constrained Application Protocol (CoAP); draft-ietf-core-coap-12
    - Core Link Format; RFC 6690
    - Observing Resources in CoAP; draft-ietf-core-observe-07
    - Blockwise transfers in CoAP; draft-ietf-core-block-10
- Additional features
    - Reverse Proxy
    - Full set of options
- CoAP Binding to ETSI M2M REST Resources

Table 4: 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 separate response (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)

Table 5: 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)

Table 6: 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)

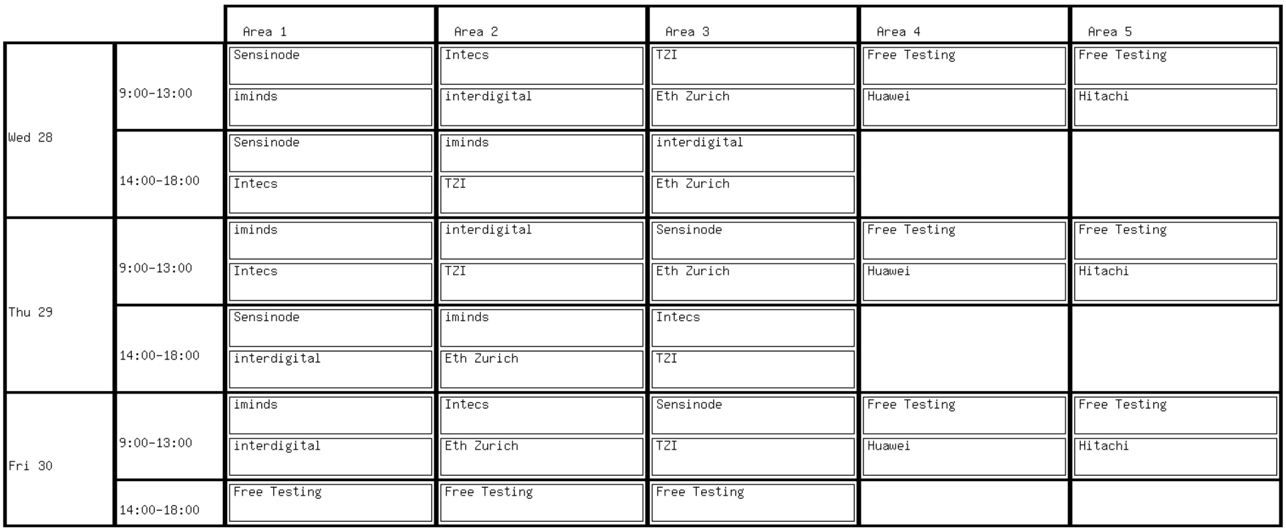
6.2 Test Scheduling

The preliminary test schedule was developed before the Plugtest and was circulated to all the participants in advance for comments. The initial test schedule allowed for each company to test against a fair number of other companies. Two companies were assigned one test slot which had duration of 4 hours. In this test slot the companies could run tests for the configurations: CompA-Client-CompB- Server and CompA-Server-CompB-Client. Up to 5 test sessions in parallel were planned.

During the test event the test schedule was updated according to the progress of the test sessions. This was done during the daily wrap-up meetings at the end of each day and during face-to-face meetings with the participants.

The figure below shows the final version of the test schedule.

Figure 1: Test Schedule



		Area 1	Area 2	Area 3	Area 4	Area 5
Wed 28	9:00-13:00	Sensinode iminds	Intecs interdigital	TZI Eth Zurich	Free Testing Huawei	Free Testing Hitachi
	14:00-18:00	Sensinode Intecs	iminds TZI	interdigital Eth Zurich		
Thu 29	9:00-13:00	iminds Intecs	interdigital TZI	Sensinode Eth Zurich	Free Testing Huawei	Free Testing Hitachi
	14:00-18:00	Sensinode interdigital	iminds Eth Zurich	Intecs TZI		
Fri 30	9:00-13:00	iminds interdigital	Intecs Eth Zurich	Sensinode TZI	Free Testing Huawei	Free Testing Hitachi
	14:00-18:00	Free Testing	Free Testing	Free Testing		

6.3 Interoperability Test Procedure

Each test was executed in the same manner as listed below:

1) Connect client and server over test network

2) Check connectivity between devices

3) Perform tests according to Plugtest Guide

    a. Check if test runs to completion

    b. Check results from an interoperability point of view:
    Is the intended result visible at the application layer?

    c. Submit the traces (packets capture) to the passive validation tool and check the result

4) Result determination and reporting

    a. Result OK: run next test

    b. Result not OK: check monitor tools to identify source of error

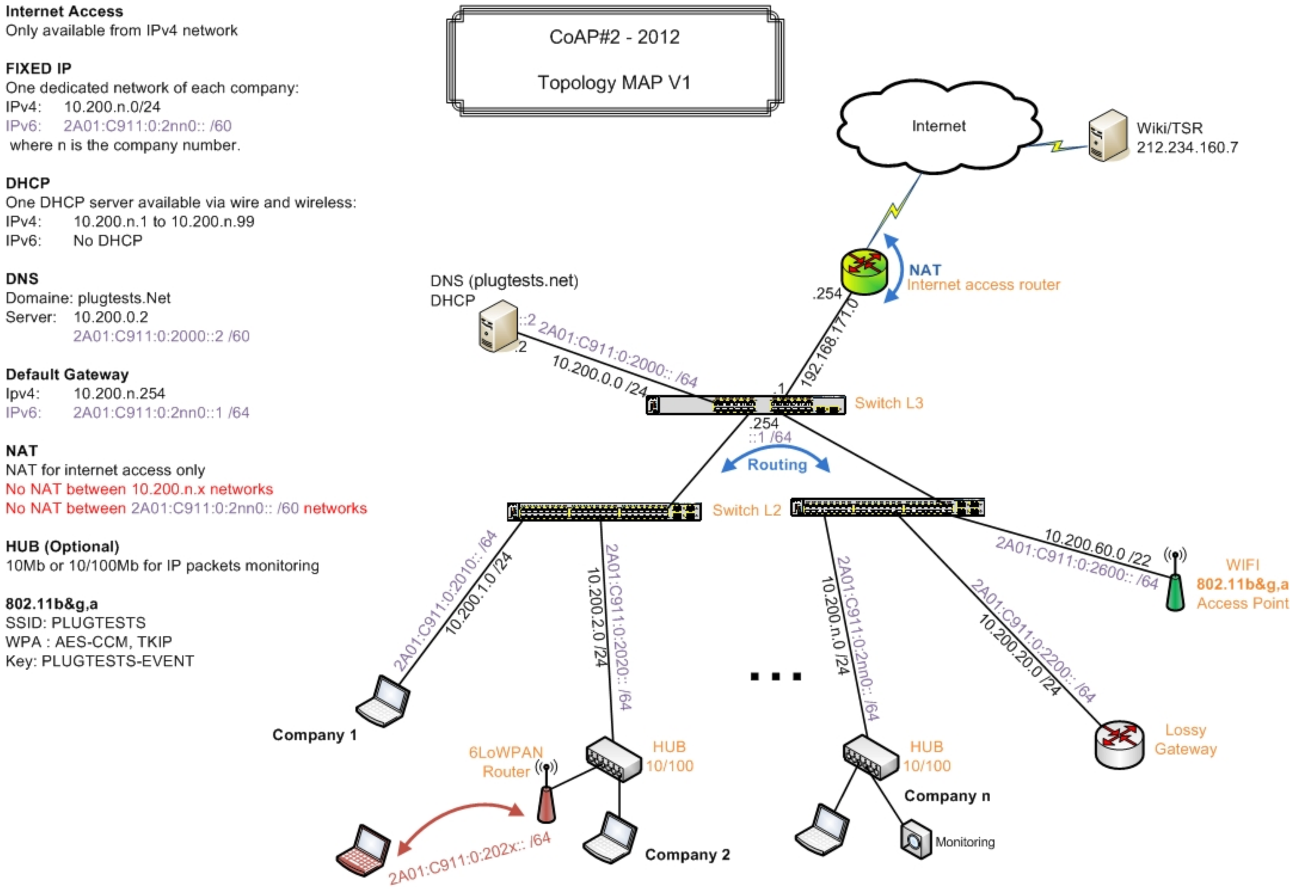
    c. Report results in ETSI Test Reporting Tool

5) Once all tests executed swap client / server roles and run all tests again

6.4 Test Infrastructure

The test infrastructure provided for the Plugtest is shown below.

Figure 2: Test Network



6.5 Tooling

6.5.1 ETSI Test Reporting Tool

The purpose of the ETSI Test Reporting Tool is to provide a means to report the test sessions. It provides statistical overviews of the test results. The graphical information in the latter section on results was created with the ETSI Test Reporting Tool. It also provides a means to create a test schedule (see section 6.2).

6.5.2 IRISA Online Trace Validation

The purpose of the passive validation tool for the CoAP protocol is to validate the traces in a capture file (in the pcap format) against the scenarios detailed in the test specification.

The tool performs a passive analysis of the network traffic recorded in the traces. It automatically detects which clients and servers are present and which scenarios are run. Then it checks the content of the exchanged messages and produces a detailed report.

The validation tool is accessible via a web browser, the traces are submitted through a web form. Additionally a shell script was provided to the participants to automate the capture and submission of traces and ease the archival of results.

All details about this tool are located on the web site:
http://www.irisa.fr/tipi/wiki/doku.php/Passive_validation_tool_for_CoAP

6.5.3 BUPT UDP Lossy Gateway

The purpose of the UDP lossy gateway is to perform packet loss in CoAP conversations according to the lossy context test descriptions defined in the plugtest guide.

The configuration of the setup is shown below:

CoAP Client ----- UDP Lossy Gateway ----- CoAP Server

Figure 3: UDP Lossy Gateway Configuration

The UDP lossy gateway assigns one listening port for each CoAP server. Thus the UDP lossy gateway provides for each CoAP server a unique lossy address.

A CoAP client that does lossy context test sends the CoAP message to the lossy address of the specified CoAP server. Then the UDP lossy gateway decides the right destination address according to the UDP socket on which the message was received.

Then the UDP lossy gateway starts a new UDP socket to communicate with the appropriate CoAP server. This UDP socket is also used for forwarding back the CoAP server's responses to the right CoAP client. The server-side communication expires after idling 5mn.

Packet loss is performed at 2 places:

- forwarding CoAP client's message to the CoAP server
- forwarding back CoAP server's message to the CoAP client

The program generates random numbers to decide whether to perform packet loss or not. A 30% packet loss rate was used for the plugtest.

6.5.4 Pre-Testing

Prior to the event, 3 companies had posted on the wiki some software(client/plugin or wireshark dissector) and also the addresses of CoAP servers, in order to enable the participants to run pre-testing. The feedback we received is that it has been appreciated and helpful for preparing the event.

7 Achieved Results

The achieved results show that all implementations have been compatible on a basic level, i.e. sent data could be decoded and interpreted properly by receivers and a vast majority of equipment performed well.

During the tests sessions capture files were produced, and uploaded to the IRISA tool. This exercise showed that conformance testing would be beneficial.

7.1 Overall Results

Due to NDA constraints, it is not possible to provide detailed results.

The figure below shows the overall result of mandatory and optional tests. In a total more than 1775 tests were executed.

The execution rate of 45.5% is a satisfying result, especially as considering the high number of tests proposed in such short event. It appears that due to lack of time or non feature support, the participants focused on mandatory tests which explains such execution rate.

Even if the test sessions were long (4 hours), most of the companies had several devices (client and servers) that of course increased the number of possible pairing combinations. Globally, the feedback that the participants gave is that the testing was very dense and they concentrated themselves on the mandatory tests.

97.8% of the test verdicts were PASS which shows a very high level of maturity of the implementations.

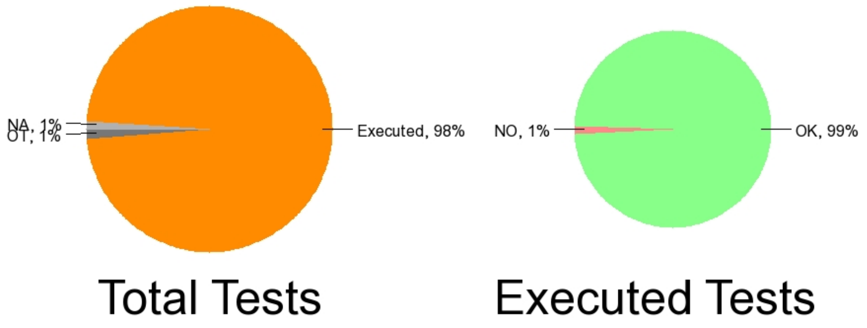


Figure 4: Overall Results

7.2 Results of mandatory tests

There were 23 mandatory tests defined which were to be executed per reported sessions. In a total 1348 mandatory tests were executed. The figures below reflect the results as described in section 7.1

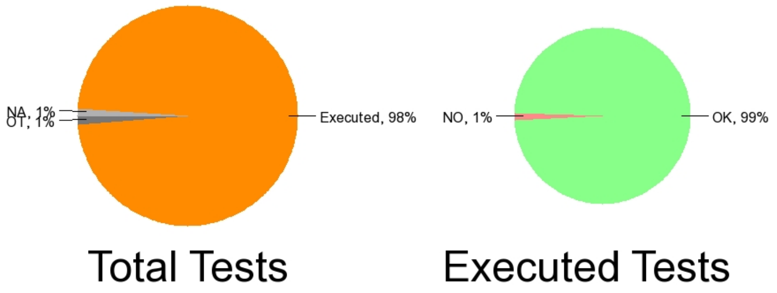


Figure 5: Results of mandatory tests

7.3 Results of optional tests

There were 29 optional tests defined which were to be executed per session. In a total 423 optional tests were executed. The figures below reflect the results as described in section 7.1. A high percentage (60.1%) of tests have been reported as “Non Applicable” and not “Out of Time”, which seems to indicate that not all implementations have fully covered all features proposed.

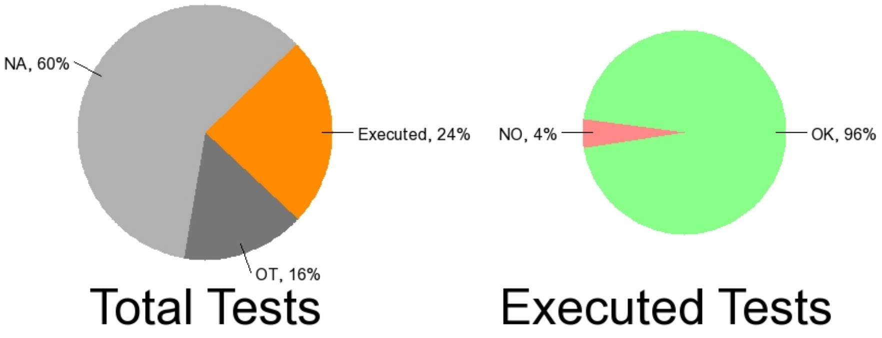


Figure 6: Results of optional tests

8 Summary of Wrap Up Sessions

8.1 IOP Issues

No real big interoperability issues were detected. The few test scenarios executions (2,2%) that led to Fail of Inconclusive verdicts were mainly due to configuration issues or implementations not implementing correctly the specification. Few issues were observed on test specifications and are described below.

8.2 Test Spec Issues

Feedback received during the Plugtest is listed here below and needs to be implemented for a next event.

3 main issues in the Test Specifications have been raised during the Wrap-up session the first day of the event. We have generated a new version v0.0.13 in the day but we opted to keep the changes as minimal as possible to avoid any confusion. Only 3 test scenarios are impacted in this new version:

- TD_COAP_CORE_22: a intermediate GET request was inserted after the first PUT request so as to retrieve the new ETag of the updated resource
- TD_COAP_LINK_05: the « if="" » pre-condition changed to « no if attribute »
- TD_COAP_OBS_02: use a new observable `/obs-non` for which the server is configured to send non-confirmable notifications.

Regarding the other OBS test scenarios, we clarified that the `/obs` resources produces confirmable notifications.

8.3 Base Specification Issues

The following two sections describe technical issues encountered during the 2nd IoT Plugtest. They are related to the current versions of the CoAP[1] and Observe[2] drafts and provide suggestions to the IETF Core working group.

8.3.1 Definition of the ETag option (draft-ietf-core-coap-12)

The current definition of the ETag option states that it is related to the target payload enclosed in the CoAP message:

```
5.10.7. ETag

The ETag Option in a response provides the current value of the
entity-tag for the enclosed representation of the target resource.
```

This definition implies that the ETag option can only be used in messages containing a payload; otherwise its content would be meaningless. This has some implications in other parts of the draft:

- Section 5.9.1.3 states that 2.03 (Valid) responses MUST contain an ETag option to identify the current valid version of the requested resource. However, it would not make sense to include a payload, since the client already knows the resource content. Also this section does not address the presence or absence of a payload in 2.03 responses.

- Sections 5.9.1.1 and 5.9.1.3 do not address the presence or absence of the ETag option in 2.01 (Created) and 2.04 (Changed) responses (which do not carry any payload). It would be worth to clarify this point since httpbis[7] explicitely allows ETag (as a MAY) in 201 and 204 responses.

The definition of ETag will need to be modified to be applicable on 2.03 response. One solution is to re-use the term "selected representation" introduced in httpbis [7].

```
8.2. Selected Representation Header Fields

We use the term "selected representation" to refer to the the current
representation of a target resource that would have been selected in
a successful response if the same request had used the method GET and
excluded any conditional request header fields.

Additional header fields define metadata about the selected
representation, which might differ from the representation included
in the message for responses to some state-changing methods.  The
following header fields are defined as selected representation
metadata:

+-------------------+-----------------------+
| Header Field Name | Defined in...         |
+-------------------+-----------------------+
| ETag              | Section 2.3 of [Part4] |
| Last-Modified     | Section 2.2 of [Part4] |
| Vary              | Section 8.2.1         |
+-------------------+-----------------------+
```

Suggestions sent to the IETF CORE WG:

- To update the definition of ETag to make it compliant payload-less messages (especially in 2.03 responses)

- To clarify that no payload should be present in 2.03 responses

- To clarify whether ETag may be included or not in 2.01 and 2.04 responses (following PUT/POST requests)

These points have been already taken into account in the new draft ietf-core-coap-13.

8.3.2 Confirmable vs. Non-confirmable notification (observe option): draft-ietf-core-observe-07

There was some confusion in the writing of the interoperability test specification for the IoT CoAP#2 Plugtest event, especially one (invalid) assumption was that a confirmable request with an observe option would produce confirmable notifications, and respectively that a non-confirmable request would produce non-confirmable notifications.

- The "Server-side requirements" section of the observe draft[3] makes clear that the decision to send a notification as a confirmable or a non-confirmable message is up to the server:

```
   4.2. Notifications

   A notification can be sent as a confirmable or a non-confirmable
   message.  The message type used is typically application-dependent
   and MAY be determined by the server for each notification
   individually.
```

- Regarding the same point the "Client-side requirement" section is rather terse:

```
   3.2. Notifications

   A notification can be confirmable or non-confirmable (i.e. sent in a
   confirmable or non-confirmable message).
```

Suggestions sent the IETF CORE WG:

- In Section 3.2: clarify that "client implementations MUST be prepared to receive each server notifications equally as a confirmable or a non-confirmable message, regardless of the message type (CON or NON) of the request and of the previous notifications".

Annex A CoAP Interoperability Test Specification

The CoAP Interoperability Test Specification, which forms parts of the present technical report, is contained in the file IoT_CoAP2_TestSpecification_13.pdf and is available in the Supplementary Information zip file provided with this document.

History

Document history		
V0.0.1	December 2012	Initial version
V0.0.3	December 2012	Updated with technical outcomes
V1.0.0	January 2013	Final version