HORIZON 2020
Information and Communication Technologies
Integrating experiments and facilities in FIRE+

Deliverable D2.1
Online interop test core enablers 1st iteration

Grant Agreement number:	**687884**
Project acronym:	**F-Interop**
Project title:	**FIRE+ online interoperability and performance test tools to support emerging technologies from research to standardization and market launch** **The standards and innovations accelerating tool**
Type of action:	**Research and Innovation Action (RIA)**
Project website address:	**www.finterop.eu/**
Due date of deliverable:	**M12**
Dissemination level:	**PU**

This deliverable has been written in the context of the Horizon 2020 European research project F-Interop, which is supported by the European Commission and the Swiss State Secretariat for Education, Research and Innovation. The opinions expressed and arguments employed do not engage the supporting parties.

Co-funded by the European Union

Co-funded by the Swiss Confederation

Document properties

Responsible partner	INRIA
Author(s)/editor(s)	**Federico Sismondi**, **César Viho**, Remy Léone, Thomas Watteyne
Version	1.0
Keywords	Interoperability Testing, Remote Testing, Online, Platform, Testing components, Test enablers

Abstract

This report corresponds to the deliverable **D2.1 – Online Interoperability test core enablers 1st iteration**. It provides the first release of the online interoperability test core enablers. It contains the first version the F-interop online remote interoperability testing framework and required key enablers needed. First version of new methods and components to simplifying online remote interoperability testing are presented.

Table of Contents

List of Figures

List of Tables

List of Acronyms

6TiSCH	IPv6 over the TSCH mode of IEEE 802.15.4e
CoAP	Constrained Application Protocol
EC	European Commission
ETSI	European Telecommunications Standards Institute
EU	European Union
GPS	Global Positioning System
HTTPS	Hypertext Transfer Protocol Secure
ICT	Information and Communication Technologies
ID	Identifier
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IoT	Internet of Things
IP	Internet Protocol
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
ISO	International Standards Organization
IT	Information Technology
MAC	Media Access Control
OS	Operating System
R&D	Research & Development
SME	Small Medium Enterprise
TAT	Test Analysis Tool
TED	Test Extended Description
TL	Task Leader
URL	Uniform Resource Locator
WP	Work Package
W3C	World Wide Web Consortium

1 Introduction

1.1 About F-Interop

F-Interop is a Horizon 2020 European Research project, which proposes to extend the European research infrastructure (FIRE+) with online and remote interoperability and performance test tools supporting emerging technologies from research to standardization and to market launch. The outcome will be a set of tools enabling:

- Standardization communities to save time and resources, to be more inclusive with partners who cannot afford travelling, and to accelerate standardization processes;
- SMEs and companies to develop standards-based interoperable products with a shorter time-to-market and significantly lowered engineering and financial overhead.

F-Interop intends to position FIRE+ as an accelerator for new standards and innovations.

1.2 Deliverable Objectives

1.2.1 Work package Objectives

- Research and develop the online remote interoperability test key enablers
- Develop the conformance test enablers
- Implement and fine tune the requested tools with a modular architecture for extensibility

1.2.2 Task Objectives

1.2.2.1 T2.1: Online interop test core enablers M1-M33 (Task Leader: Inria)

Work. The main objective of this task is to define and implement the components of the F-Interop online remote interoperability-testing framework. This includes: the cloud-based interoperability test script repository (as well as its management), the test servers and test suites automation, as well as the libraries, adapters, API and hardware interfaces, and its reporting capability. This task includes developing new methods simplifying online remote interoperability testing. This task also considers associated security and authentication issues.

Roles: Inria will lead the task, and will integrate contributions from ETSI, UL and EANTC.

Outcome: All components and key enablers needed for online remote interoperability testing.

T2.2: Complementary conformance test enablers M1-M33 (Task Leader: Inria)

Work. Conformance is a pre-requisite for interoperability. This task will provide the enablers for online remote conformance testing which complement the key enabled developed in task T2.1. We will develop new methods and/or adaptations of existing conformance testing tools to take into account the specific case of interacting online and remotely with the implementation under test (IUT). Examples include sniffers platforms, tools for measuring end-to-end latency (for example based on GPS synchronization), a protocol dissector.

Roles: Inria will lead the task, and will integrate contributions from ETSI, UL and EANTC.

Outcome: Additional components needed for online remote conformance testing. These components will complement the key enablers developed in task T2.1.

1.2.3 Deliverables Objectives and Methodology

The work to be done to provide Interoperability test core enablers and Complementary test enablers is strongly linked, most of the components in this deliverable **D2.1 – Online Interoperability test core enablers 1st iteration** are also part of those that has to be provided for the deliverable **D2.2 - Complementary conformance test enablers 1st iteration**. Consequently, references to sections that are common with this deliverable D2.1 will be provided in the deliverable D2.2. Meanwhile, part of the work that is specific to this deliverable D2.1 are given in separate sections hereafter.

1.2.3.1 Deliverables Objectives

The deliverable **D2.1 – Online Interoperability test core enablers 1st iteration** is the first release of the online interoperability test core enablers. This report contains the first version of the F-interop online remote interoperability testing framework and the required key enablers needed. First version of new methods to simplify online remote interoperability testing are presented.

1.2.3.2 Deliverables Methodology

We have considered the two types of testing tools in WP2 that the F-Interop has to deal with: ***Online Interoperability testing*** and ***Online conformance testing***. We decided to select some of the targeted emerging IoT technologies that cover as many layers/aspects as possible of the IoT protocol stack. After internal discussion, we decided to focus first on the following protocols: 6TiSCH and CoAP. For each of these two types of testing and these two selected protocols (6TiSCH and CoAP), we have investigated the state of the art (existing methods and tools) for testing, and we have studied and compared existing IoT related testing solutions and tools. These studies helped us starting the discussion on what a user might expect from the F-Interop-Platform. Based on the test scenarios that have been developed and used during previous and recent interoperability face-to-face (F2F) interoperability testing events, we started studying what is needed for doing the same but in an online and remote manner. Based on the lessons learnt from all the considered uses cases of 6TiSCH and CoAP, we identified the set of actions that any user (called F-Interop-User) has to perform in order to execute remotely interoperability and conformance tests using the F-Interop-Platform. This set of actions is called an "**F-Interop session**" and it was summarized in the Table 1 of the deliverable D1.1 which is attached below.

Step	Action	Description
0	FI-User authentication and authorization. IUT registration / identification	FI-User authenticates in a secure way (prior FI-User registration needed) in FI-Platform. FI-User needs to be authorized to use FI-Platform resources. FI-User identifies which IUT he/she will test (prior IUT registration needed).
1	Test suites discovery and selection	FI-User starts by discovering the available test suites and by selecting the one he/she wants to execute.
2	Resource description	FI-User specifies/selects resources in the F-Interop-Platform that are needed for his/her F-Interop session including the location models[1], testing tools, libraries, etc. During this phase FI-Platform may request information from FI-User or provide information to FI-User for a coherent selection of the required resources.
3	Resource reservation	The resources selected in the previous step are actually reserved.
4	Resource provisioning, configuration and session	The instantiation of the F-Interop-Platform resources that

[1] Location models are the different configurations for the location of components of the test. These will be defined in D1.3.1. [6]

	setup	fit best with the FI-User needs is done.
5	**Test execution**	The online F-Interop test campaign is launched and the selected (executable) test suites are executed against the IUTs.
6	**Results analysis and report**	Test execution information is analysed. The test results and verdicts are provided together with explanations in case of FAIL or INCONCLUSIVE verdicts or something wrong happened. A report can be provided under request in case for example the FI-User wants to apply for a certification/labelling program.
7	Session storage	Storage of the F-Interop session information (Session-id, User-id, FI-User's IUT-id, IUTs' version, test description, test version, testing tool, test log and results, etc.). This has to remain accessible beyond the F-Interop session for the involved parties.

Table 1 - F-Interop Session

The work reported in this document, that corresponds to the deliverable D2.1, focused more specifically on the **step 5 (test execution)** and the **step 6 (result analysis and report)** of a F-Interop session. To better understand these steps we provide in the following section a F-Interop User's journey for remote online conformance and interoperability testing.

1.2.3.2.1 Executing remotely conformance and interoperability in F-Interop

The two figures below show the core idea of what we intend to provide to allow executing remotely and online conformance and interoperability test in the F-Interop context.

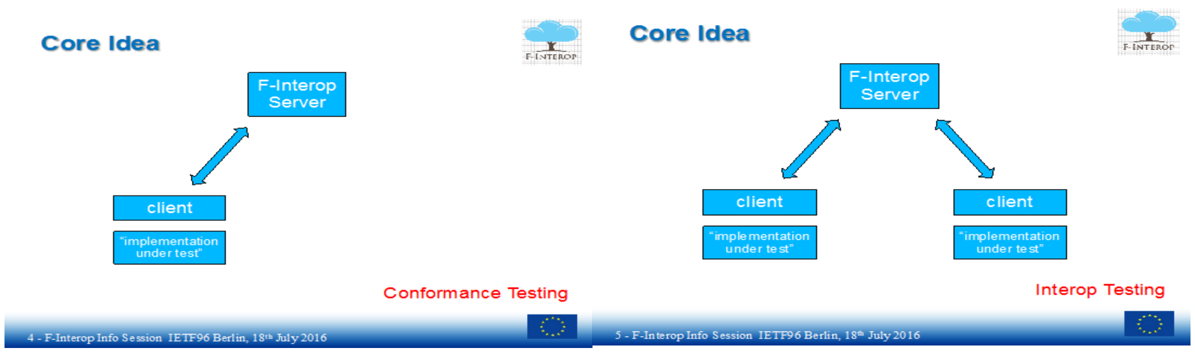
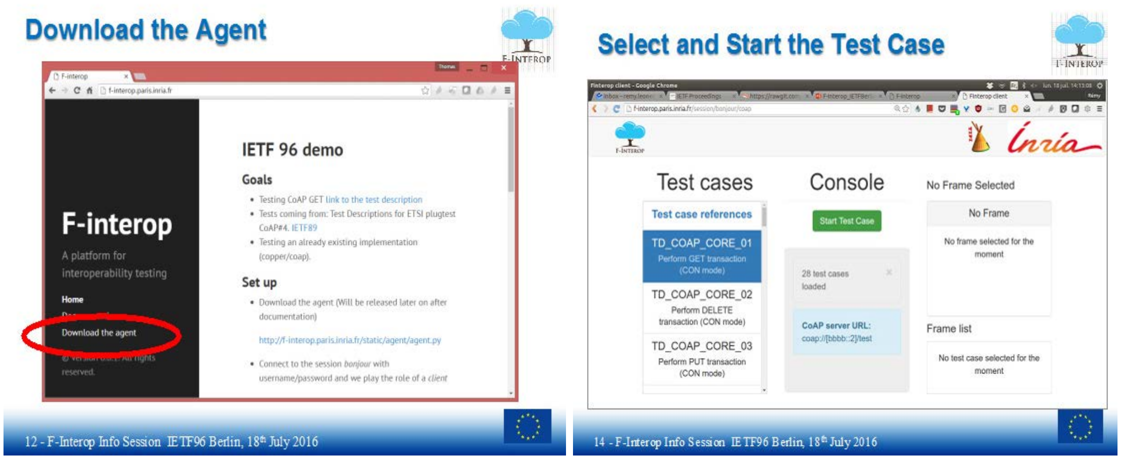

Core Idea
F-Interop Server
client
"implementation under test"
Conformance Testing
4 - F-Interop Info Session IETF96 Berlin, 18th July 2016
Core Idea
F-Interop Server
client
"implementation under test"
client
"implementation under test"
Interop Testing
5 - F-Interop Info Session IETF96 Berlin, 18th July 2016

Figure 1 Executing conformance and interoperability in F-Interop

For the conformance testing on the left of Figure 1, the testing tool on a F-Interop server executes remotely the tests against a distant IUT. A client or agent will be in charge of managing the interaction between the testing tool and the IUT. In the case of interoperability on the right side of the Figure 1, the interoperability testing tool is on the F-Interop server and is executed remotely against two or more distant IUTs. In both cases, the only thing that a F-Interop User has to do is to download the agent, to connect to the F-Interop server, to select the tests to be executed and to launch them.


Download the Agent

F-interop

A platform for interoperability testing

Home

Download the agent

reserved.

IETF 96 demo

Goals

- Testing CoAP GET link to the test description
- Tests coming from: Test Descriptions for ETSI plugtest CoAP#4. IETF89
- Testing an already existing implementation (copper/coap).

Set up

- Download the agent (Will be released later on after documentation)

  http://f-interop.paris.inria.fr/static/agent/agent.py

- Connect to the session *bonjour* with username/password and we play the role of a *client*

12 - F-Interop Info Session IETF96 Berlin, 18th July 2016

Select and Start the Test Case

Test cases | Console | No Frame Selected

Test case references

TD_COAP_CORE_01 Perform GET transaction (CON mode)

TD_COAP_CORE_02 Perform DELETE transaction (CON mode)

TD_COAP_CORE_03 Perform PUT transaction (CON mode)

Start Test Case

28 test cases loaded

CoAP server URL: coap://[bbbb::2]/test

No Frame

No frame selected for the moment

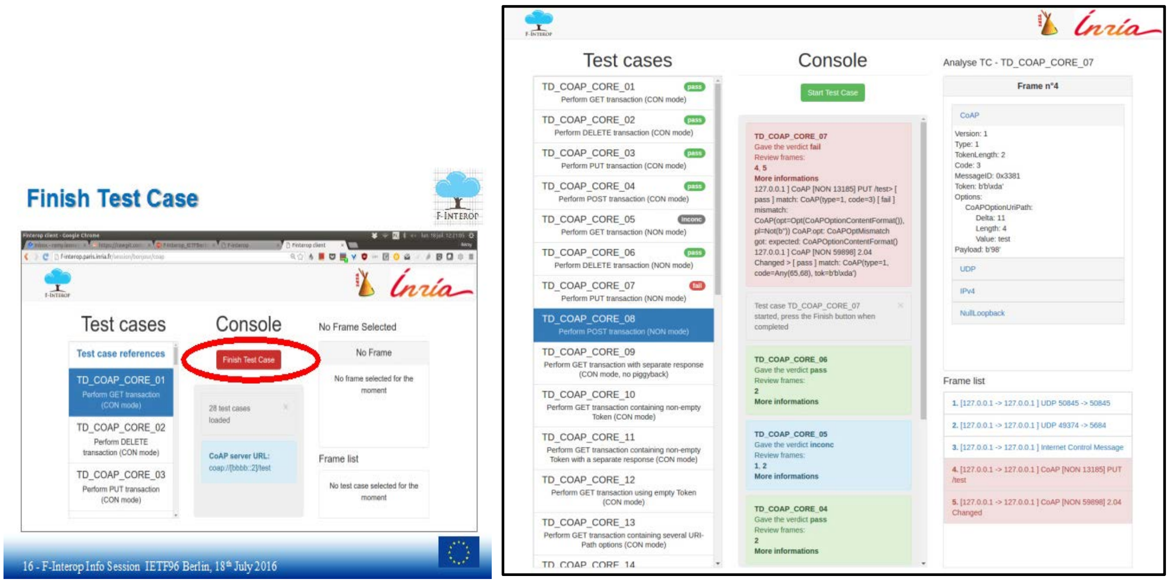
Frame list

No test case selected for the moment

14 - F-Interop Info Session IETF96 Berlin, 18th July 2016

Figure 2 Launching remote test execution in F-Interop

All the selected are executed and as soon as their executions are finished, the F-Interop User gets the results as indicated on the right side of the Figure 3.


Finish Test Case

Test cases | Console | No Frame Selected

Test case references

TD_COAP_CORE_01 Perform GET transaction (CON mode)

TD_COAP_CORE_02 Perform DELETE transaction (CON mode)

TD_COAP_CORE_03 Perform PUT transaction (CON mode)

Finish Test Case

28 test cases loaded

CoAP server URL: coap://[bbbb::2]/test

No Frame

No frame selected for the moment

Frame list

No test case selected for the moment

16 - F-Interop Info Session IETF96 Berlin, 18th July 2016

Test cases

- TD_COAP_CORE_01 pass — Perform GET transaction (CON mode)
- TD_COAP_CORE_02 pass — Perform DELETE transaction (CON mode)
- TD_COAP_CORE_03 pass — Perform PUT transaction (CON mode)
- TD_COAP_CORE_04 pass — Perform POST transaction (CON mode)
- TD_COAP_CORE_05 inconc — Perform GET transaction (NON mode)
- TD_COAP_CORE_06 pass — Perform DELETE transaction (NON mode)
- TD_COAP_CORE_07 fail — Perform PUT transaction (NON mode)
- TD_COAP_CORE_08 — Perform POST transaction (NON mode)
- TD_COAP_CORE_09 — Perform GET transaction with separate response (CON mode, no piggyback)
- TD_COAP_CORE_10 — Perform GET transaction containing non-empty Token (CON mode)
- TD_COAP_CORE_11 — Perform GET transaction containing non-empty Token with a separate response (CON mode)
- TD_COAP_CORE_12 — Perform GET transaction using empty Token (CON mode)
- TD_COAP_CORE_13 — Perform GET transaction containing several URI-Path options (CON mode)
- TD_COAP_CORE_14

Console

Start Test Case

```
TD_COAP_CORE_07
Gave the verdict fail
Review frames:
4, 5
More informations
127.0.0.1 ] CoAP [NON 13185] PUT /test> [
pass ] match: CoAP(type=1, code=3) [ fail ]
mismatch:
CoAP(opt=Opt(CoAPOptionContentFormat()),
pl=Not(b'')) CoAP.opt: CoAPOptMismatch
got: expected: CoAPOptionContentFormat()
127.0.0.1 ] CoAP [NON 59898] 2.04
Changed > [ pass ] match: CoAP(type=1,
code=Any(65,68), tok=b'b\xda')
```

Test case TD_COAP_CORE_07 started, press the Finish button when completed

```
TD_COAP_CORE_06
Gave the verdict pass
Review frames:
2
More informations
```

```
TD_COAP_CORE_05
Gave the verdict inconc
Review frames:
1, 2
More informations
```

```
TD_COAP_CORE_04
Gave the verdict pass
Review frames:
2
More informations
```

Analyse TC - TD_COAP_CORE_07

Frame n°4

```
CoAP
Version: 1
Type: 1
TokenLength: 2
Code: 3
MessageID: 0x3381
Token: b'b\xda'
Options:
    CoAPOptionUriPath:
        Delta: 11
        Length: 4
        Value: test
Payload: b'98'
UDP
IPv4
NullLoopback
```

Frame list

```
1. [127.0.0.1 -> 127.0.0.1 ] UDP 50845 -> 50845
2. [127.0.0.1 -> 127.0.0.1 ] UDP 49374 -> 5684
3. [127.0.0.1 -> 127.0.0.1 ] Internet Control Message
4. [127.0.0.1 -> 127.0.0.1 ] CoAP [NON 13185] PUT /test
5. [127.0.0.1 -> 127.0.0.1 ] CoAP [NON 59898] 2.04 Changed
```

Figure 3 Test results: verdicts and logs in F-Interop

To allow remote online conformance and interoperability testing, we need to define and develop all components that compose the F-Interop testing architecture and framework.

1.2.3.2.2 Towards a modular and generic architecture for remote online interoperability and conformance testing

Internal discussions with partners involved in the WP2 as well as with the whole consortium helped us in identifying key components for online remote testing described in detail in deliverable D1.1 (see table 5 in section 5.3 of the deliverable D1.1) and led to the definition of the first version of the F-Interop architecture. This architecture has been discussed in detail allowing agreement on several points including the initial implementation design. It is agreed that the overall architecture view will have clear indication on the interactions and the interfaces/APIs/standards to be used among the various components. The architecture will be as modular and as generic as possible, to ease the extension towards new standards and services and to enable a coherent, consistent and interoperable developments of components by the various tasks. In the same way, three main location models (see definition in deliverable D1.1) have been identified to be considered as starting point.

The AMQP (Advanced Message Queuing Protocol) standard is agreed as being an interesting solution on the tester side. RabbitMQ[1] messaging tool implements AMQP and enables differentiated ACL security/authorization profiles. AMQP bindings are available in several languages. They can be used for remote interaction with the IUTs. Orchestration and distinct channels can be used for distinct testing tools.

The architecture presented in Figure 4 below tackles the complexities of doing test in this online and remote manner. The following sections describe the architecture of the testing tools, its components and other auxiliary components for performing interoperability (and conformance) tests.

2 F-Interop interoperability core enablers

2.1 F-Interop testing architecture and components

In this section the F-Interop testing architecture is presented in Fig. 1. The components of this architecture are responsible for managing the testing infrastructure necessary, including provisioning the underlying network, capturing trace, starting/stopping the different tests, and reporting the verdicts. Through standard security mechanisms, the architecture ensures the authentication of the different users, and the confidentiality of test results.

2.1.1 The “Event Bus” Software Design Pattern

The F-Interop architecture is composed of different components exchanging messages through an “Event Bus”. All communication is done through this mechanism, including control messages, raw data packets and logs. We use RabbitMQ as the underlying message-passing mechanism. It acts as a secure message broker between all the components through encrypted channels.

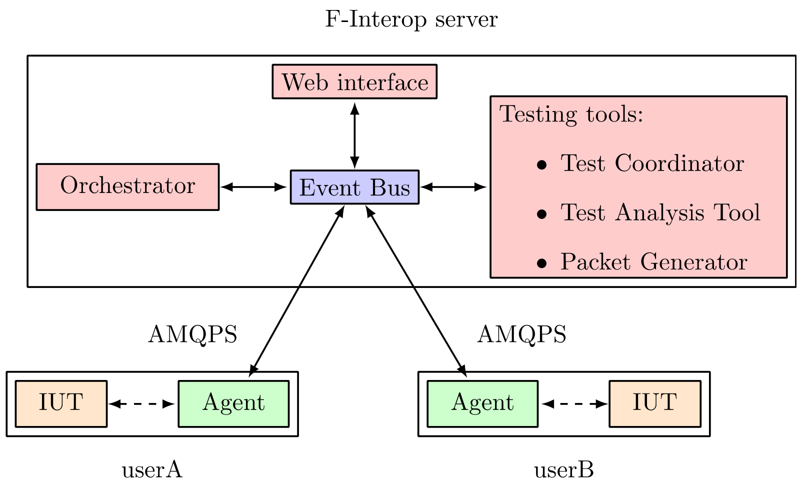


Figure 4 – F-Interop remote interoperability testing architecture

Each message contains a routing key and a topic which indicates how to route this message to the relevant input queues of the components. Messages are of two types: ***control plane*** and ***data plane***. The ***control plane*** messages relate to the management of an ongoing test session: e.g. start a sniffer, signal the start/end of a test case, etc. The ***data plane*** messages contain the raw data exchanged between the IUTs.

A F-Interop-User conducts remote interoperability tests in independent sessions. We use the virtual host mechanism of RabbitMQ[1] to ensure isolation between concurrent sessions.

This architecture is modular and scalable by design. Components can be added/deleted from the event bus without requiring further coordination. Different components can be run on different (virtual) machines to ensure scalability. Different components can be written in different programming languages.

2.2 Components description

2.2.1 Agent: Connecting F-Interop-Users to the F-Interop Platform

An **agent** is a program that a F-Interop-User downloads from the F-Interop website, and which allows him/her to connect an IUT to the F-Interop server. Communication between the agent and the server is authenticated and secure. Through the agent, the F-Interop server can (remotely) interact with the IUT, for example by changing configuration or injecting packets. Similarly, the agent reports events to the server, such as sniffed packets.

2.2.2 The Orchestrator

The **orchestrator** plays a purely administrative role: it monitors the users that are connected, launch required processes, provision the message broker and starts/stops the test sessions. It does so by spawning/killing the processes of the different components connected to the event bus. It uses the supervisor[2] process control system.

2.2.3 Testing tool

The **testing tool** handles the components throughout a F-Interop session, it can be started once the different users are connected and the necessary components are provisioned by the orchestrator. The role of the testing tool is to generate verdicts that corresponds to test cases. While the F-Interop platform does not impose a particular design for the testing tool. It operates as a black box which emits coordination messages towards agents, and generates test verdicts. It is typically composed of a test coordinator, a test analysis tool and a packet generator (mainly for conformance testing).

2.2.3.1 Test Coordinator

The **test coordinator** is the component that glues all parts together within the testing tool. Its purpose is to coordinate the test cases execution during a F-Interop session. It is a generic executor of the test case, although, at the bootstrapping phase, it retrieves a F-Interop session context from the test orchestrator.

Some of the most common actions handled by this component are:

- SNIFFER: start, stop, retrieve captured files.
- GUI: Messages related to the execution (e.g. “execute test case 2 step 5”)
- AGENT: configure interface, start interface, launch automated step, etc.
- TAT: analyze capture, provide dissection of a frame.

This component is protocol agnostic; once instantiated it gets the specific test context (test suite, analysis mode, etc.) from the test orchestrator and feeds from the test extended descriptions (TED yaml files); see below.

2.2.3.2 Test Extended Description

Each **Test Extended Description** (TED) describes in detail each steps and actions of a test case execution. TEDs are meant to be easy to write using YAML language, and they are written by the F-Interop contributors. This is the code that describes the configurations and the steps of each test case. It is a translation of a test case of the test descriptions (TD) into machine understandable language. Just like the TDs, the TEDs describe the set of steps that need to be executed.

Typically, there are 3 types of steps:

- STIMULI: an action for stimulating the IUT (e.g. sending a message X to the IUT).

- CHECK: the action of validating the communication (e.g. check that the field X is equal to value Y).
- VERIFY: the action of verifying that an IUT behaves correctly (e.g. verify that resource A updated its value to B).

An example of a TED is provided in section 6.1

2.2.3.3 Test Analysis Tool

The **Test Analysis Tool (TAT)** is the component that performs the verification of traces during a F-Interop session. F-Interop provides TATs for different protocols, which run after the message exchange is finished.

The TAT issues three types of verdicts:

- PASS when test purpose of the test case is verified, meaning that nothing incorrect/invalid has been observed
- FAIL when at least something incorrect/invalid has been observed during the test case execution
- INCONCLUSIVE when the behavior of the IUTs are authorized but does not apply to the one described in the conformance (resp. interoperability) test purpose.

The architecture supports TATs which perform both step-by-step analysis or post mortem analysis. TATs are created both by the F-Interop core team and by external contributors. The F-Interop API specification defines the format of the messages a TAT will receive from the Event Bus, and the format of the messages it can produce.

2.2.3.4 Packet Generator

Mainly in the context of conformance tests, a **packet generator** component is in charge of building and generating packets that are sent to the IUT.

2.2.4 Web Interface

The F-Interop **Web interface** allows the user to select a test description from a list of available tests, start the execution of the test description and follow the execution of the different test cases. In some cases, the web interface can request the user to take some action (e.g. switch off a node). The web interface also allows the user to retrieve the test report. The web interface communicates with the rest of the system by sending/receiving message over the Event Bus.

3 Status of work

As it has been previously mentioned, the work related to T2.1 and T2.2 has been carried out in parallel due to the mutualized F-Interop testing architecture. In the following sub-sections, we present some indicators on the status of work of the first iteration of Online Interoperability test core enablers corresponding to the deliverable D1.1. As said previously, these components are based on CoAP interoperability tests suite experiences.

3.1 Status of work for T2.1 – Deliverable D2.1

The work accomplished for the first iteration of the T2.1 includes modularization of the testing tools components, interfaces specification, and the development of a first version of the core components (Agent, Orchestrator, Test Coordinator, Web Interface, Test Analysis Tool) that allow to perform remote online interoperability testing for CoAP; see the figure below.

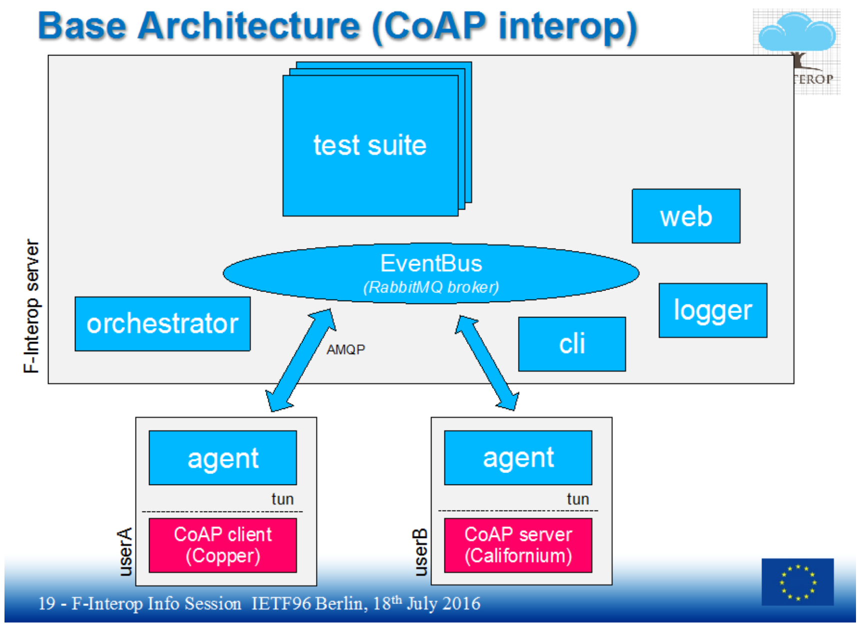


Figure 5 Base architecture for CoAP remote interoperability testing

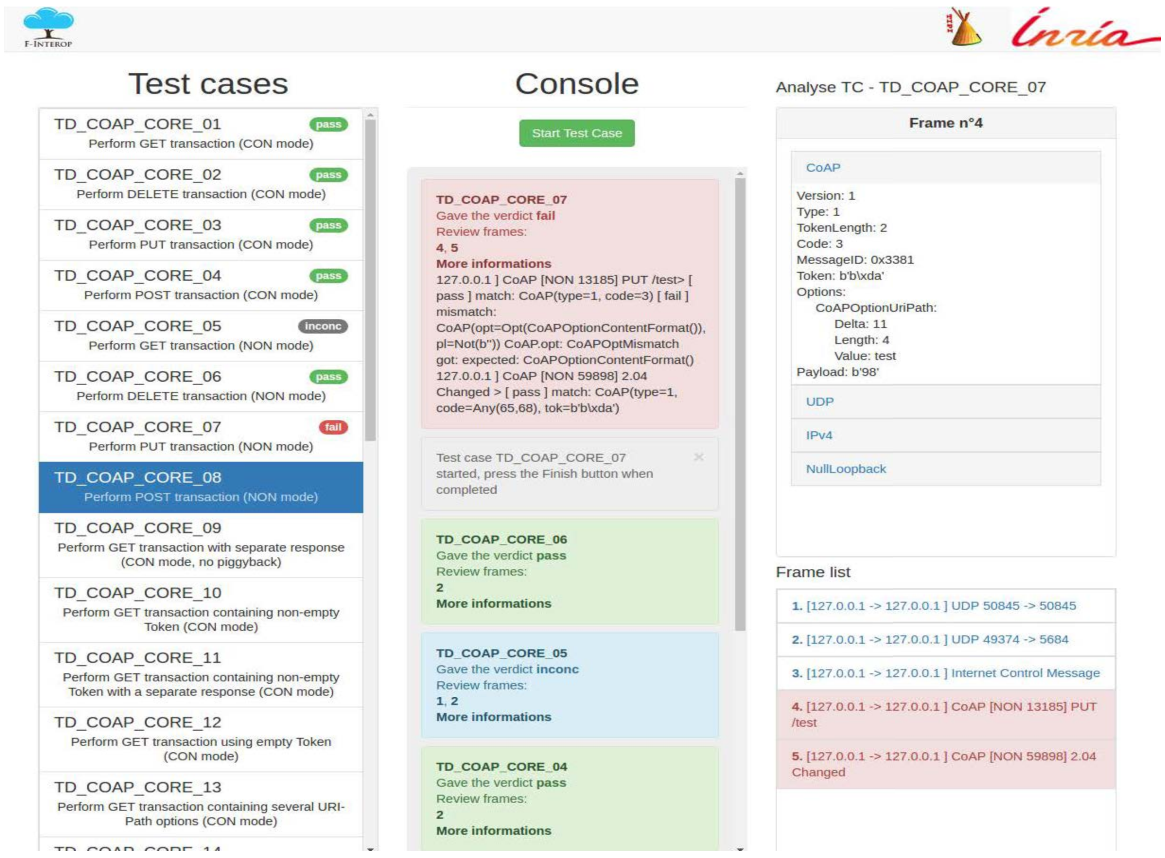
One of the major challenges throughout WP2 was to implement an architecture which favors modularity and components reuse in a protocol agnostic manner. This is the reason why a big effort has been made on defining the interfaces and the specification of the messages exchanged between components within the testing tool block (aligned with work in T1.1, T1.2 and T1.3 from WP1). This increases the possibility of re-using some components when implementing new testing tools for new protocols.

Besides working on the identification and specification these interfaces, F-interop partners have focused in developing two key components of the testing tool architecture, which are test analysis tool (see Section 2.2.3.3), and the test coordinator (see Section 2.2.3.1). These two blocks are still under development and efforts are being made to keep these components as generic and protocol agnostic as possible. At the same time CoAP extensions of the components are being developed for verification of their correct behavior. For this purpose, three test cases have been implemented for CoAP remote interoperability tests. An example of Interoperability test description can be seen in Annex 6.1. These included the development of three TEDs (test extended description) on the Test Coordinator side (see an example in Annex 6.2), and their corresponding three Test Scripts which run on top of the Test Analysis Tool implementation (see Annex 6.3).

Along with this work, a basic web version of a GUI (see Figure 6) has been developed for helping the F-Interop User running the tests throughout the test session execution. The GUI development doesn't concern WP2, but this minimal version helps developers testing and debugging their implementations and is also useful for demo purposes.

To summarize, the work accomplished for the first iteration of the T2.1 includes the development of a first version of the following components (Agent, Orchestrator, testing tool, Test coordinator, Web interface, Packet generator, Test analysis tool) that allow to perform remote online interoperability testing for CoAP.

- Definition of interfaces between components of testing tools (aligned with work in T1.1, T1.2 and T1.3 from WP1)
- First implementation of the interfaces using the event bus messaging pattern for components such as test coordinator and agents.
- First implementation of a generic test coordinator (see Section 2.2.3.1)
    - Test Extended Descriptions written for 3 CoAP interoperability test cases
- First implementation of the agent component (see Section 2.2.1)
    - Tested for CoAP remote interoperability tests.
- Implementation of a test analyzer tool (see Section 2.2.3.3)
- A basic Web GUI for demo purposes (GUI development work is out of WP2 scope)
- First implementation of CoAP remote interoperability test suite which was presented at IETF 96 meeting in Berlin and W3C TPAC meeting in Lisbon.



TD_COAP_CORE_01 pass
Perform GET transaction (CON mode)

TD_COAP_CORE_02 pass
Perform DELETE transaction (CON mode)

TD_COAP_CORE_03 pass
Perform PUT transaction (CON mode)

TD_COAP_CORE_04 pass
Perform POST transaction (CON mode)

TD_COAP_CORE_05 inconc
Perform GET transaction (NON mode)

TD_COAP_CORE_06 pass
Perform DELETE transaction (NON mode)

TD_COAP_CORE_07 fail
Perform PUT transaction (NON mode)

TD_COAP_CORE_08
Perform POST transaction (NON mode)

TD_COAP_CORE_09
Perform GET transaction with separate response (CON mode, no piggyback)

TD_COAP_CORE_10
Perform GET transaction containing non-empty Token (CON mode)

TD_COAP_CORE_11
Perform GET transaction containing non-empty Token with a separate response (CON mode)

TD_COAP_CORE_12
Perform GET transaction using empty Token (CON mode)

TD_COAP_CORE_13
Perform GET transaction containing several URI-Path options (CON mode)

TD_COAP_CORE_07
Gave the verdict **fail**
Review frames:
4, **5**
More informations
127.0.0.1] CoAP [NON 13185] PUT /test> [pass] match: CoAP(type=1, code=3) [fail] mismatch: CoAP(opt=Opt(CoAPOptionContentFormat()), pl=Not(b'')) CoAP.opt: CoAPOptMismatch got: expected: CoAPOptionContentFormat() 127.0.0.1] CoAP [NON 59898] 2.04 Changed > [pass] match: CoAP(type=1, code=Any(65,68), tok=b'b\xda')

Test case TD_COAP_CORE_07 started, press the Finish button when completed

TD_COAP_CORE_06
Gave the verdict **pass**
Review frames:
2
More informations

TD_COAP_CORE_05
Gave the verdict **inconc**
Review frames:
1, **2**
More informations

TD_COAP_CORE_04
Gave the verdict **pass**
Review frames:
2
More informations

Analyse TC - TD_COAP_CORE_07

Frame n°4

CoAP
```
Version: 1
Type: 1
TokenLength: 2
Code: 3
MessageID: 0x3381
Token: b'b\xda'
Options:
    CoAPOptionUriPath:
        Delta: 11
        Length: 4
        Value: test
Payload: b'98'
```
UDP
IPv4
NullLoopback

Frame list

1. [127.0.0.1 -> 127.0.0.1] UDP 50845 -> 50845
2. [127.0.0.1 -> 127.0.0.1] UDP 49374 -> 5684
3. [127.0.0.1 -> 127.0.0.1] Internet Control Message
4. [127.0.0.1 -> 127.0.0.1] CoAP [NON 13185] PUT /test
5. [127.0.0.1 -> 127.0.0.1] CoAP [NON 59898] 2.04 Changed

Figure 6 - Web GUI for running interop tests

4 Conclusion and next steps

This document presented the work related to the task T2.1 that has been carried out in parallel with the task 2.2 due to the mutualized F-Interop testing architecture. We present some indicators on the status of work of the first iteration of Online Interoperability test core enablers corresponding to deliverable D1.1. We first presented the components of the F-Interop testing architecture that are responsible for managing the testing infrastructure necessary, including provisioning the underlying network, capturing trace, starting/stopping the different tests, and reporting the verdicts. The outcome of the work achieved in this period is the development of the first version of these components based on CoAP interoperability tests suite experiences. It includes the development of a first version of the following components: The Agent, Orchestrator, testing tool, Test coordinator, Web interface, Packet generator, Test analysis tool). They allowed performing remote online interoperability testing for CoAP.

The work to be done in next steps includes:

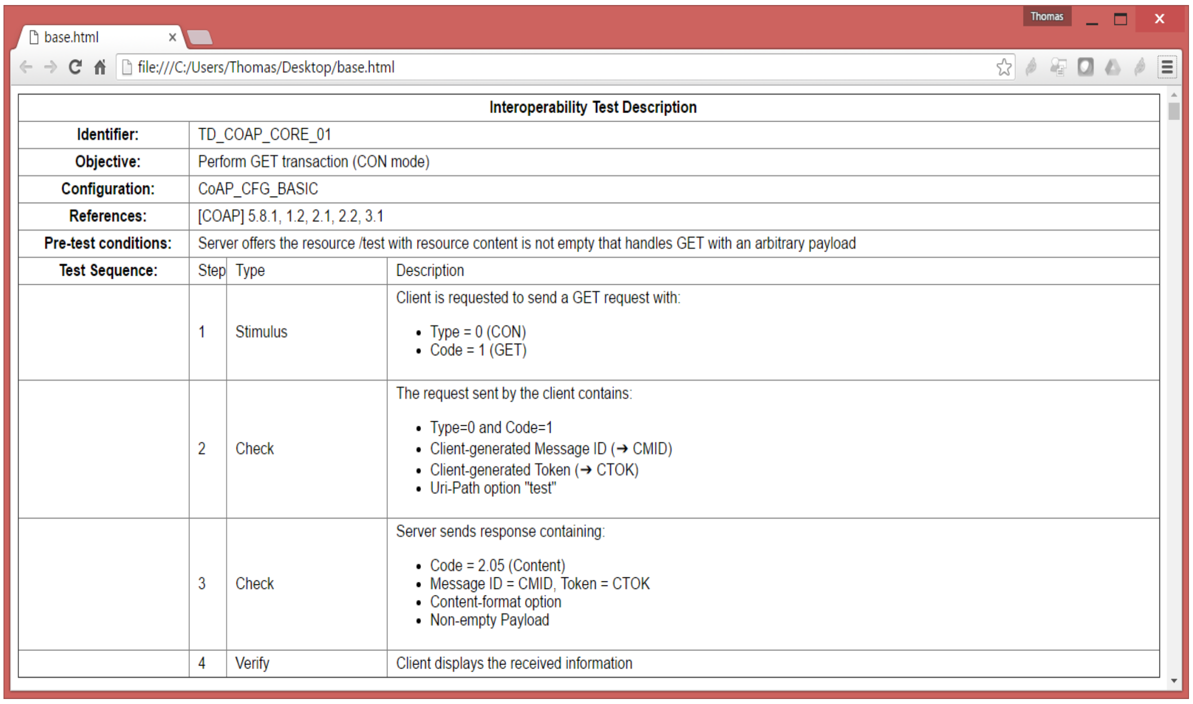
- Completing the development of the components of the online remote testing architecture
- Implementation more tests for CoAP
- Integration of the testing architecture to the testbeds (FIT IoT-Lab, iMinds, etc.) allowing executing remotely online interoperability with components on those testbeds
- Extending testing tools for including Web of Things (WoT) interoperability testing tools.

5 References

[1] https://www.rabbitmq.com/
[2] http://supervisord.org/
[3] https://tools.ietf.org/html/draft-wang-6tisch-6top-protocol-00
[4] https://www.ietf.org/mail-archive/web/6tisch/current/pdfgDMQcdCkRz.pdf
[5] https://openwsn.atlassian.net/wiki/display/OW/OpenVisualizer
[6] http://zeromq.org

6 Annex

6.1 An example of CoAP interoperability test description from ETSI plugtest CoAP#4, IETF89 (London)



base.html
file:///C:/Users/Thomas/Desktop/base.html

Interoperability Test Description			
Identifier:	TD_COAP_CORE_01		
Objective:	Perform GET transaction (CON mode)		
Configuration:	CoAP_CFG_BASIC		
References:	[COAP] 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 • Client-generated Message ID (➔ CMID) • Client-generated Token (➔ CTOK) • Uri-Path option "test"
	3	Check	Server sends response containing: • Code = 2.05 (Content) • Message ID = CMID, Token = CTOK • Content-format option • Non-empty Payload
	4	Verify	Client displays the received information

6.2 An example of Test Extended Description (YAML file)

```
testcase_id: TD_COAP_CORE_01
uri : http://f-interop.paris.inria.fr/tests/TD_COAP_CORE_01
```

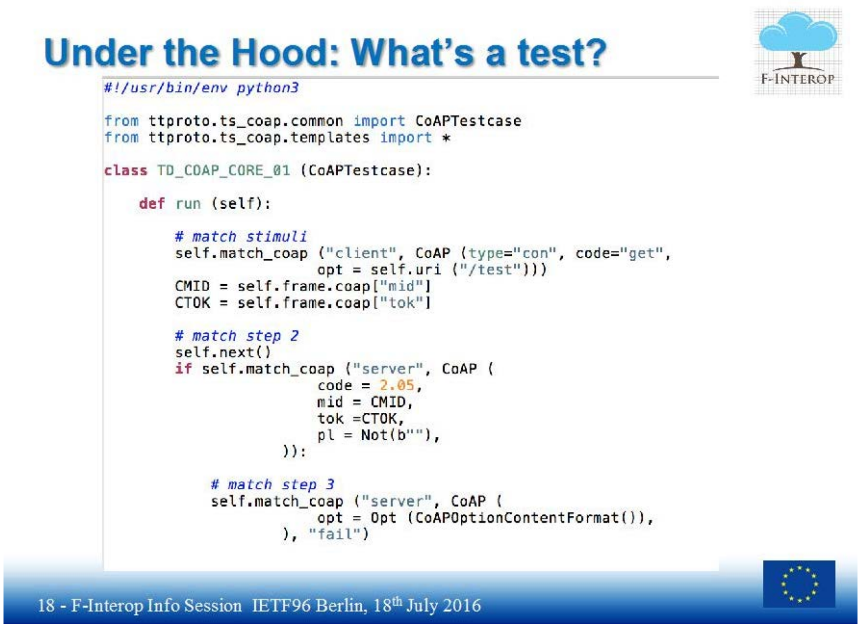
```
configuration: CoAP_configuration_BASIC
objective: Perform GET transaction(CON mode)
pre_conditions: Server offers the resource /test with resource content is not empty that handles GET with an
arbitrary payload
references: '[COAP] 5.8.1, 1.2, 2.1, 2.2, 3.1'
sequence:
  - step_id: 'TD_COAP_CORE_01_v01_step_01'
    type: stimuli
    iut : coap_client
    description:
      - Client is requested to send a GET request with
      - Type = 0(CON)
      - Code = 1(GET)

  - step_id: TD_COAP_CORE_01_v01_step_02
    type: check
    description:
      - The request sent by the client contains
      - Type=0 and Code=1
      - Client-generated Message ID(\u2794 CMID)
      - Client-generated Token(\u2794 CTOK)
      - Uri-Path option "test"

  - step_id: TD_COAP_CORE_01_v01_step_03
    type: check
    description:
        - Server sends response containing
        - Code = 2.05(Content)
        - Message ID = CMID, Token = CTOK
        - Content-format option
        - Non-empty Payload

  - step_id: TD_COAP_CORE_01_v01_step_04
    type: verify
    iut: coap_client
    description:
        - Client displays the received information
```

6.3 An example of CoAP Interoperability Test Script



Under the Hood: What's a test?

```
#!/usr/bin/env python3

from ttproto.ts_coap.common import CoAPTestcase
from ttproto.ts_coap.templates import *

class TD_COAP_CORE_01 (CoAPTestcase):

    def run (self):

        # match stimuli
        self.match_coap ("client", CoAP (type="con", code="get",
                        opt = self.uri ("/test")))
        CMID = self.frame.coap["mid"]
        CTOK = self.frame.coap["tok"]

        # match step 2
        self.next()
        if self.match_coap ("server", CoAP (
                        code = 2.05,
                        mid = CMID,
                        tok =CTOK,
                        pl = Not(b""),
                    )):

            # match step 3
            self.match_coap ("server", CoAP (
                        opt = Opt (CoAPOptionContentFormat()),
                    ), "fail")
```

18 - F-Interop Info Session IETF96 Berlin, 18th July 2016