



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

Deliverable D1.4
Final architecture design

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: 31/07/2018

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.



Document properties

Responsible partner	imec
Author(s)/editor(s)	Brecht Vermeulen (imec), Thijs Walcarius (imec), Wim Van de Meerssche (imec), all partners
Version	1
Keywords	Final Architecture, SDK, APIs

Abstract

This deliverable is the second of two deliverables in the task 1.3 Architecture and FIRE+ integration design (D1.3 was the first version of the architecture). The requirements studied in task 1.1 and 1.2 were used as input, together with the first version of the architecture and the feedback of WP2, WP3 and WP4 during the project. We describe first the different location models (location of devices and users) that we will support in F-interop, followed by the current high level architecture of components in F-Interop and integration with FIRE+ testbeds. After that, we go into more details on the specific interaction between the components. This interaction is based on the AMQP message protocol and we have defined the message formats, functioning as an API when contributors want to add functionality to F-Interop.

The deliverable is also meant as part of the software development kit (SDK) towards (open-call) contributors.

Table of Contents

Table of Contents	3
List of Figures	4
List of Acronyms	5
1 Introduction	8
1.1 About F-Interop	8
1.2 Deliverable Objectives	8
1.2.1 Work package Objectives.....	8
1.2.2 Task Objectives	8
1.2.3 Deliverable Objectives and Methodology.....	8
2 Location models	9
2.1 Introduction	9
2.2 Symbols used	9
2.3 Location models with a single user	9
2.4 Location models with 2 or more users	11
3 High level architecture	12
3.1 High level architecture without testbeds	12
3.2 High level architecture with testbeds	13
3.3 Deploy a sandbox environment on testbeds	15
4 Detailed architecture of a single test implementation	16
4.1 AMQP message bus	16
4.2 Test Coordinator	16
4.3 Test Analysis Tool (TAT)	17
4.4 Protocol Dissector	17
4.5 Agent	17
4.6 Packet generator	18
4.7 APIs in a single test	18
5 Envisioned interaction for an end-user	19
6 Envisioned interaction for a contributor and SDK	20
7 Conclusion	21
8 Annex: Espec for sandbox	22

List of Figures

- Figure 1: High level architecture without testbeds 12
- Figure 2: High level architecture with testbeds 14
- Figure 3: jFed GUI for deploying sandbox..... 15
- Figure 4: Pointing jFed to the base of the SDK/ESpec 15
- Figure 5: Detailed architecture of a single test 16

List of Acronyms

AMQP	Advanced Message Queuing Protocol
API	Application Programming Interface
CA	Consortium Agreement
CoAP	Constrained Application Protocol
ComSoc	Communications Society
DESCA	Development of a Simplified Consortium Agreement
DHCP	Dynamic Host Configuration Protocol
DHT	Distributed Hash Tables
DNS	Domain Name System
DNSSec	Domain Name System Security Extensions
DPA	Data Protection Authorities
DPO	Data Protection Officer
DUT	Device Under Test
EC	European Commission
ENISA	European Union Agency for Network and Information Security
ETSI	European Telecommunications Standards Institute
EU	European Union
FP7	Seventh Framework Programme
GA	Grand Agreement
GA	General Assembly
GPS	Global Positioning System
HTTPS	Hypertext Transfer Protocol Secure
ICT	Information and Communication Technologies
ID	Identifier
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IERC	European Research Cluster on the Internet of Things
IETF	Internet Engineering Task Force
IoT	Internet of Things
IP	Internet Protocol
IPC	Intellectual Property Committee
IPM	IPR Monitoring and Exploitation Manager
IPR	Intellectual Property Rights
IPSEC	Internet Protocol Security
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
ISO	International Standards Organization
ISP	Internet Service Provider
IT	Information Technology
ITU	International Telecommunication Union
IUT	Implementation Under Test
KPI	Key Performance Indicator
LSPI	Legal, Security and Privacy Issues

MAC	Media Access Control
MSc	Master of Science
M2M	Machine to Machine
OASIS	Organization for the Advancement of Structured Information Standards
OECD	Organization for Economic Cooperation and Development
OS	Operating System
OSN	Online Social Network
PC	Project Coordinator
PCAP	Packet Capture
PCP	Partner Contact Person
PDPO	Personal Data Protection Officer
PERT	Program Evaluation Review Technique
PhD	Doctor of Philosophy
PM	Person Month
PMB	Project Management Board
PPR	Periodic Progress Report
PRAAT	Privacy Risk Area Assessment Tool
P&T	Post & Telecom
QoS	Quality of Service
RAND	Reasonable and Non Discriminatory
RFC	Request For Comments
RSpec	Resource Specification
R&D	Research & Development
SDK	Software Development Kit
SME	Small Medium Enterprise
SMS	Short Message Service
SOTA (or SoA)	State Of the Art
SSL	Secure Sockets Layer
TBaaS	TestBed as a Service
TC	Technical Coordinator
TCP	Transmission Control Protocol
TL	Task Leader
TLS	Transport Layer Security
Tor	The Onion Router
TRL	Technology Readiness Level
UK	United Kingdoms
UN	United Nations
UNCTAD	United Nations Conference on Trade and Development
UPRAAT	Universal Privacy Risk Area Assessment Tool
URL	Uniform Resource Locator
US	United States
VoIP	Voice over Internet Protocol
WES	Women's Engineering Society
WiTEC	Women in science, Engineering and Technology
WoT	Web of Trust
WP	Work Package

WPL	Work Package Leader
W3C	World Wide Web Consortium
XML	Extensible Markup Language
YAML	YAML Ain't Markup Language, human friendly data serialization

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

WP1 has the following goals:

- Analyze and specify the online testing tools requirements
- Analyze and specify personal data protection and security requirements
- Design and specify the F-Interop architecture

1.2.2 Task Objectives

Task 1.3 in WP1 has the specific goal of defining the architecture and the integration with FIRE+ testbeds. For this, a milestone MS2 about the initial architecture agreement was reached at month 9 of the project, the deliverable D1.3 was delivered at M12 and a milestone MS3 was delivered at M24 for an architecture update based on the progress in the project and this document delivers the final MS4 and D1.4 with the final architecture at M33 of the project.

1.2.3 Deliverable Objectives and Methodology

The requirements studied in task 1.1 and 1.2 were used as input, together with the first version of the architecture and the feedback of WP2, WP3 and WP4 during the project. First, we will highlight the current high level architecture of components in F-Interop and integration with FIRE+ testbeds. After that, we go into more details on the specific interaction between the components. This interaction is based on the AMQP message protocol and we have defined the message formats, functioning as an API when contributors want to add functionality to F-Interop.

The deliverable is also meant as part of the software development kit (SDK) towards (open-call) contributors.

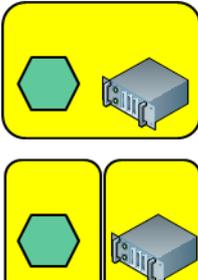
2 Location models

2.1 Introduction

To start, we identified all location models (which device and tool are located where) that F-Interop should support for its goal of developing a new experimental platform for online interoperability tests and validation tools, remote compliance and conformance tests, scalability tests, Quality of Service (QoS) tests, SDN/NFV interoperability tools, online privacy test tools, energy efficiency tools.

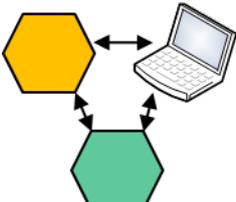
2.2 Symbols used

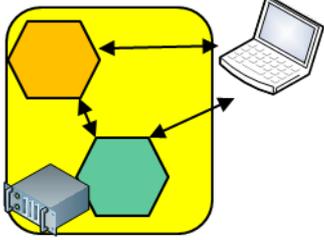
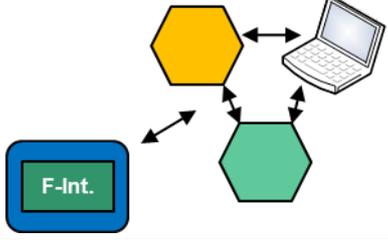
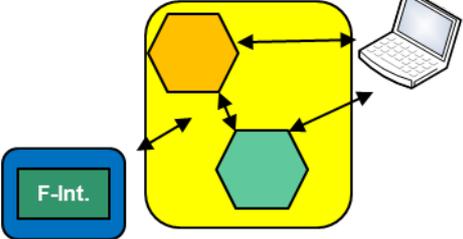
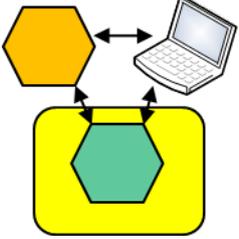
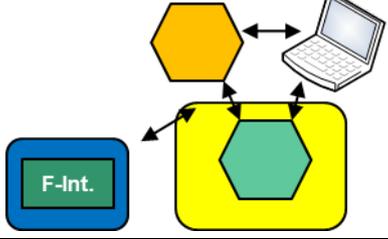
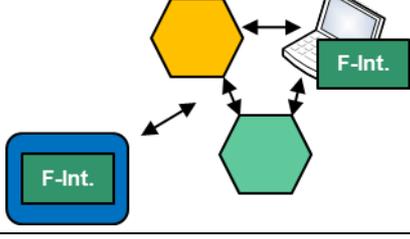
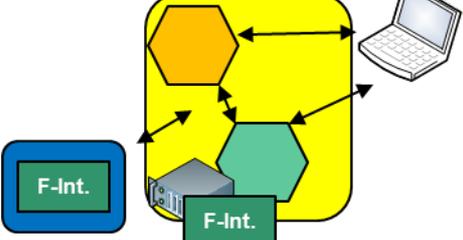
The following symbols will be used:

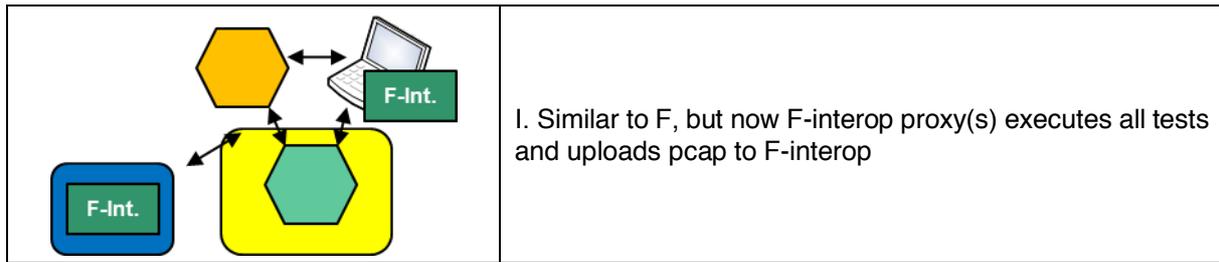
	User laptop
	Device(s) under test
	Other device(s) needed for test
	F-Interop central services
	<p>Testbed with devices (e.g. IoT devices, servers, virtual machines, network devices, ...). The second picture shows that also multiple testbeds can be involved, e.g. a Cloud testbed runs the test suite, while an IoT testbed is used for the IoT devices. The testbeds can be interesting when they offer devices that the user has not available on his desk or when more devices are needed for a test (e.g. scalability) then locally available or when you want to remotely collaborate with other users or when you want to do multiple tests or developments in parallel.</p>

2.3 Location models with a single user

We will start off illustrating the different location models we foresee to be supported by the F-Interop architecture and tools, in case of a single end-user, e.g. in case of conformance testing or in case of interoperability testing with known other devices.

	<p>A. User downloads all necessary tools and software from F-interop website and executes everything locally and processes the interaction locally (e.g. for development of new tests)</p>
---	--

	<p>B. User downloads all necessary tools and software from F-interop website and executes everything remotely in a testbed and processes the interaction locally or in the testbed (e.g. for development of new tests)</p>
	<p>C. User downloads all necessary tools and software from F-interop website and executes all locally and uploads the interaction capture (e.g. pcap file) to F-interop (e.g. User wants to orchestrate his own interactions)</p>
	<p>D. User downloads all necessary tools and software from F-interop website and executes all remotely in a testbed and uploads the interaction capture (e.g. pcap file) to F-interop (e.g. User wants to orchestrate his own interactions, but on devices he has not at his desk)</p>
	<p>E. User downloads all necessary tools and software and tests local DUT (Device Under Test) against device in testbed (or in central platform) and processes the interaction capture locally</p>
	<p>F. User downloads all necessary tools and software and tests local DUT against device in testbed (or in central platform) and uploads the interaction to F-interop for analysis</p>
	<p>G. Similar to C, but now a F-interop proxy(s) executes all tests and uploads the interaction to F-interop (e.g. Automatic test with minimum interaction of user)</p>
	<p>H. Similar to D, but now F-interop proxy(s) runs in remote testbed and executes all tests and uploads interaction to F-interop (e.g. Automatic remote test)</p>



Of course, in the F-Interop project different technologies are addressed. Not all those technologies are able to be tested in all proposed ways. E.g. CoAP is an application layer protocol that runs over IP and can thus be transported over the internet for e.g. scenario E and F above. On the other hand, 6TISCH is closer to link layer and so the devices should be in the same room for most of the tests.

2.4 Location models with 2 or more users

While the previous section talked about a single user testing against a golden implementation or device (for conformance testing) or another implementation or devices at his desk or on a testbed (for interop or performance testing, etc), it is also possible that 2 or more users test their own implementation against each other. We consider only 2 users for this kind of interop tests at this moment.

The setups are very similar to the ones listed in the previous section, but now e.g. the orange and green devices are configured and controlled by 2 different users. Especially on the scenario and interaction side, this has some implications, but F-Interop has developed its tools immediately with these cases in mind, meaning that the single user case is in fact a special case of the 2 user case. A single user has then to do all handling instead of 2 users, or in case of full automation (the F-Interop goal), has to run a proxy per device.

3 High level architecture

3.1 High level architecture without testbeds

This section will describe the high level architecture for F-Interop tests without any testbeds involved.

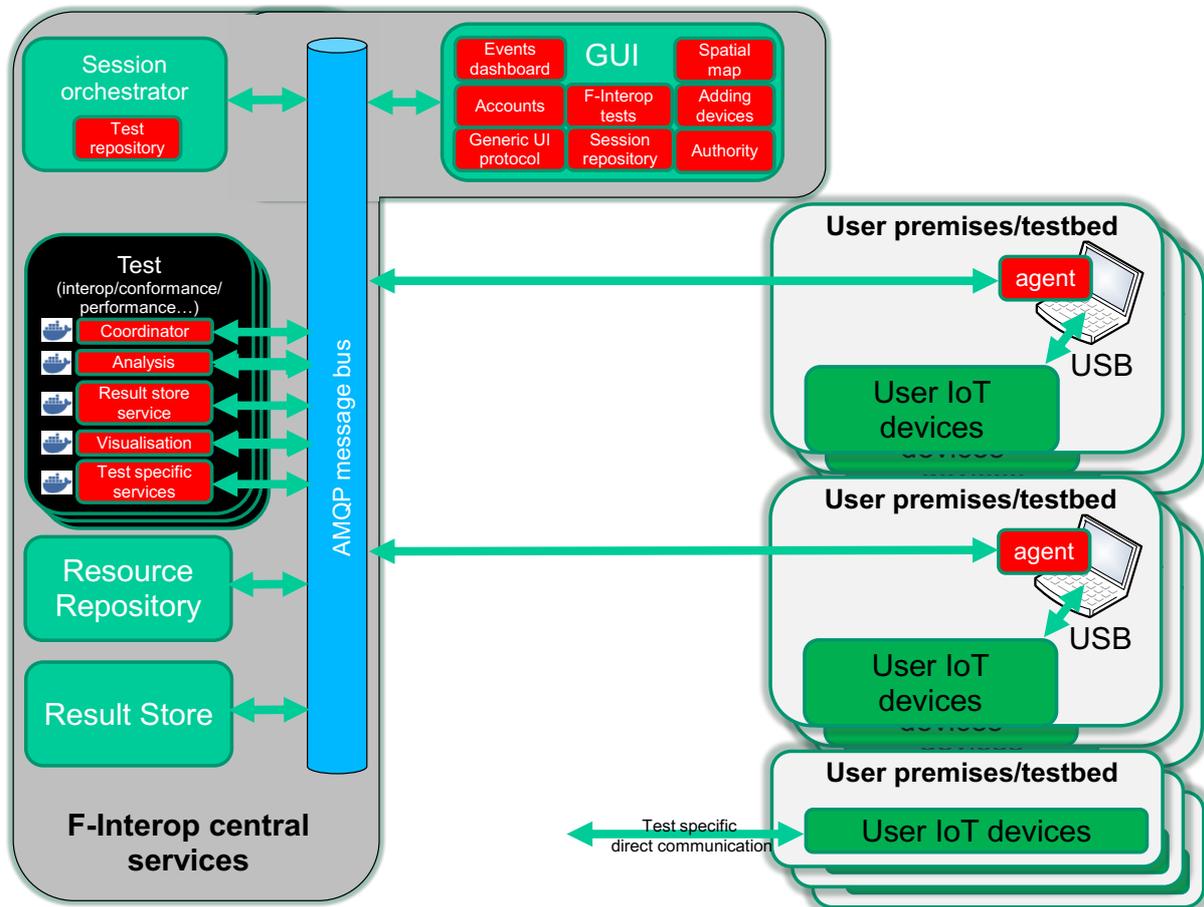


Figure 1: High level architecture without testbeds

Figure 1 shows the components that are involved at the central F-Interop services side and the user premises side (this figure shows 2 remote users doing interop tests, but the other location models are easy to derive). We identify the following components:

- The **graphical user interface (GUI)** is the first contact point for an end-user and makes it possible to register for an account, list all the F-interop tests available and register your own devices (if any). The registration of your own device can be for a single test or if you want to contribute this device so other people can also test against it, this is also possible. Other functionality of the GUI includes a session repository (to return to existing sessions e.g.), a generic UI protocol under the hood (for easy integration with new testing tools), an authority (X.509 certificates) for testbed resources and a Spatial map component for showing ongoing tests on a map. Finally, an event dashboard allows users to interact online.
- The **resource repository** contains a list of all devices and their properties that can be used for F-interop tests. The resource repository communicates with the GUI.
- The **session orchestrator** plays an administrative role: it monitors the users that are connected, activates the AMQP sessions and starts/stops the test sessions. It contains also a test repository with an overview of all available tests.

- **Each Test suite** has multiple components (which are instantiated in docker containers each being able to communicate with the AMQP message bus):
 - **The test coordinator and analysis component** is the base of an individual (interop, conformance, performance, ..) test and does first the orchestration (what should be done next according the test script, e.g. reset device) and afterwards the analysis (e.g. by analysing a pcap file which was sniffed during the interaction). Per type of test, this orchestration and analysis components have a different implementation.
 - **The result store service** interacts with the Result Store to store the specific results of this specific test.
 - Optionally there is a **test specific visualisation tool** (to show the results graphically) and **other test specific services** (e.g. for performance testing, multiple clients can be launched in other docker containers to perform the test)
- The **agent** is a software component developed by F-Interop as well. It runs near devices (e.g. at the user premises) to communicate with the orchestration and analysis components. It can do different tasks depending on the test: e.g. sniff all traffic and send a pcap to the analysis step, or e.g. full automatic orchestration of the device during the test). As indicated in the picture, for some tests, this agent is not needed/used (e.g. for performance testing).
- When the test is done, the results are stored in the central **result store** for future referral. The test results are only accessible by the people taking part in the test.

For the communication between the components, AMQP (Advanced Message Queuing Protocol, www.amqp.org) has been selected as a message bus protocol. It is used for both control message interaction as for the transmission of wrapped data packets between devices.

3.2 High level architecture with testbeds

Figure 2 starts from the basic architecture from the previous section, but adds testbed devices in the picture. The figure illustrates the setup when a contributor wants to extend an existing test, add a new test or debug in detail what goes on during testing, while using one or more devices at his desk as well during the test (development).

Through the jFed GUI of Fed4FIRE (<https://jfed.ilabt.imec.be>) and a specific F-interop Espec (Experiment specification) a sandbox environment can be deployed on testbed resources. The contributor has full access to the virtual machines and software and as such can work on the orchestration, analysis, AMQP messaging and result storage of a test. Also for performance testing, this can be handy as you have dedicated performance on the bare metal or virtual machines on the testbeds, not hindered by other users.

On the right side, it is shown that also the IoT devices themselves can be hosted on the testbed, which is only a small addition to the Espec.

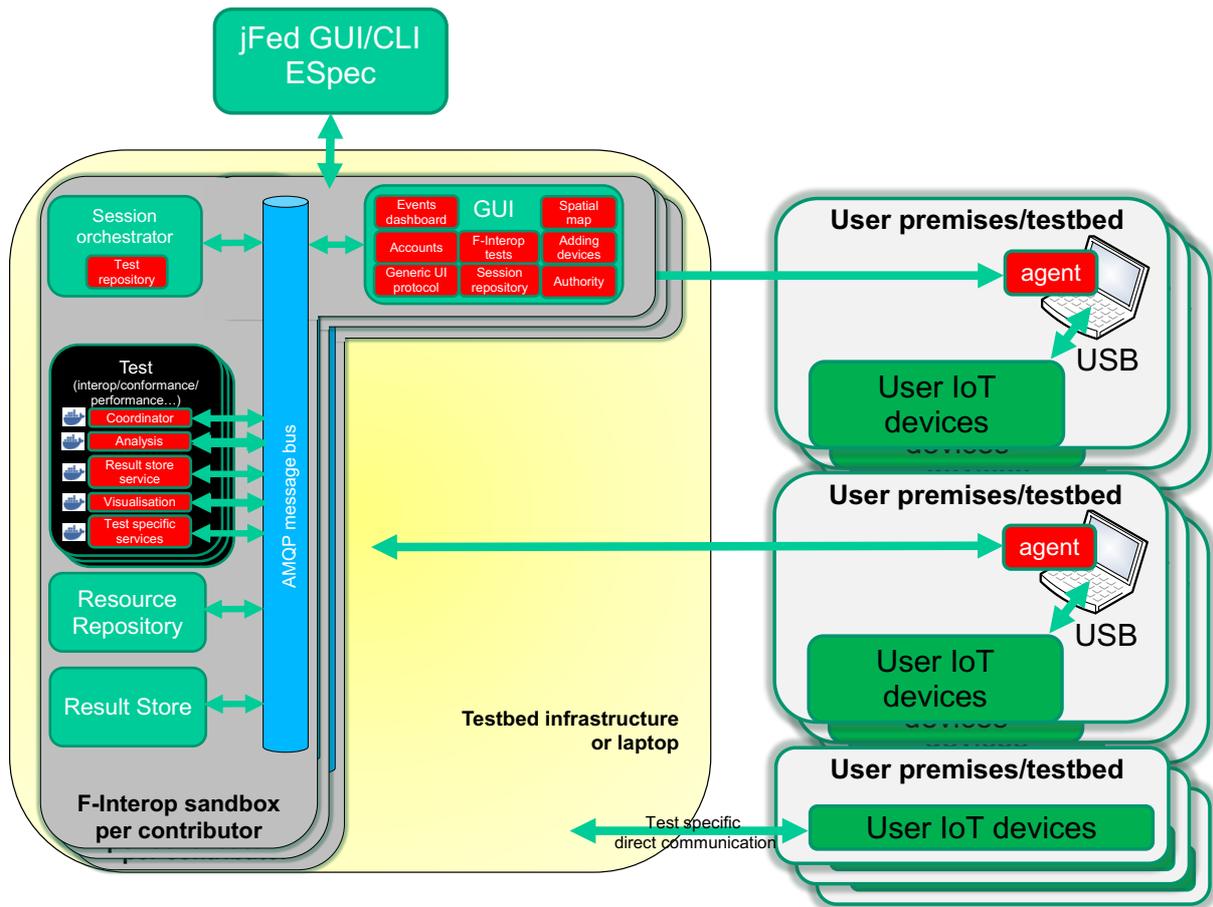


Figure 2: High level architecture with testbeds

The way in which the F-Interop tools and tests are packaged is with Ansible (<https://www.ansible.com>). In Ansible you can specify Ansible Playbooks described in YAML for to automate tasks. In this way we define Playbooks for installing AMQP, F-Interop orchestrators and analysis tools, and it becomes very easy to deploy the F-Interop framework automatically.

This is also the way to extend the F-Interop test portfolio: create and develop new tools, but package them in Ansible for easy installation and replication. Thus, it is also easy to track and use different versions of a tool.

The other components are similar to the previous section.

Figure 2 shows also how it looks like when multiple users in parallel deploy the F-Interop framework on testbeds. Each user has its own instance, with its own AMQP message bus and completely securely separated from the other instances. On the other hand, the user has full access and as such full flexibility on his instance.

3.3 Deploy a sandbox environment on testbeds

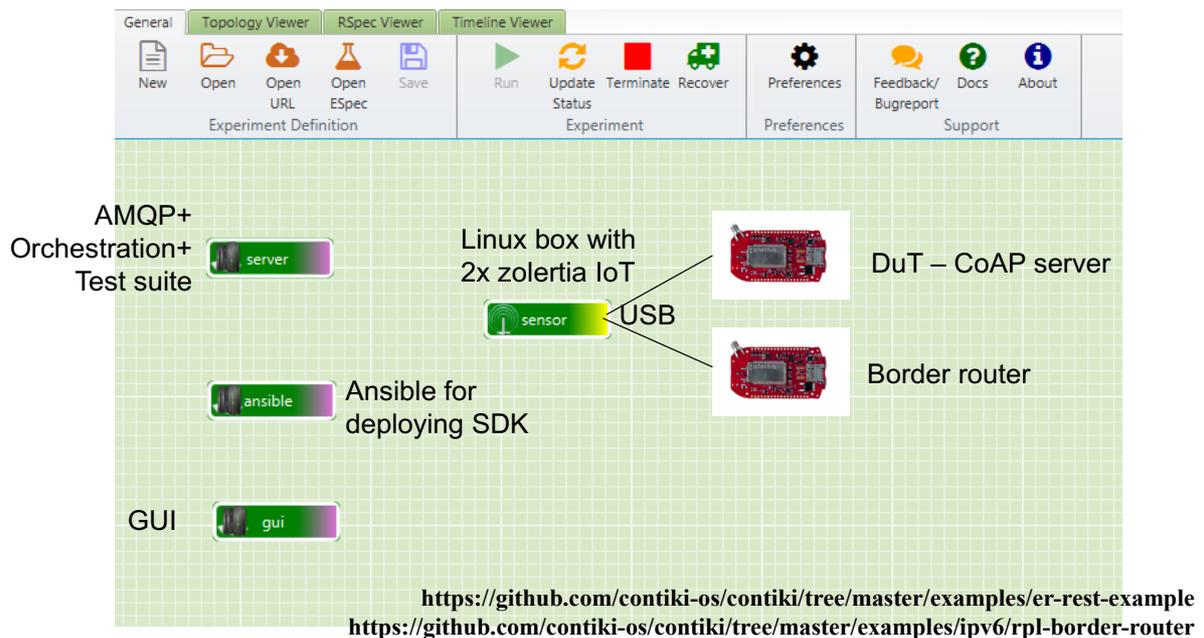


Figure 3: jFed GUI for deploying sandbox

Figure 3 shows a screenshot of jFed (<https://jfed.ilabt.imec.be>) loaded with an Espec for deploying a full sandbox environment, including a linux box on the testbed with 2 Zolertia Re-motes attached.

The howto describing how to use this, is documented at <https://gitlab.f-interop.eu/f-interop-contributors/sdk/wikis/home>. The Espec is a folder structure (or zip file containing the same folder structure), which can be found at <https://gitlab.f-interop.eu/f-interop-contributors/sdk>. The file [experiment-specification.yml](#) is the base file. jFed can be pointed to the folder structure or to the zip file containing everything, and will launch the sandbox environment. Nightly testing of the SDK can be found at <http://fedmon.fed4fire.eu/history/1327>.

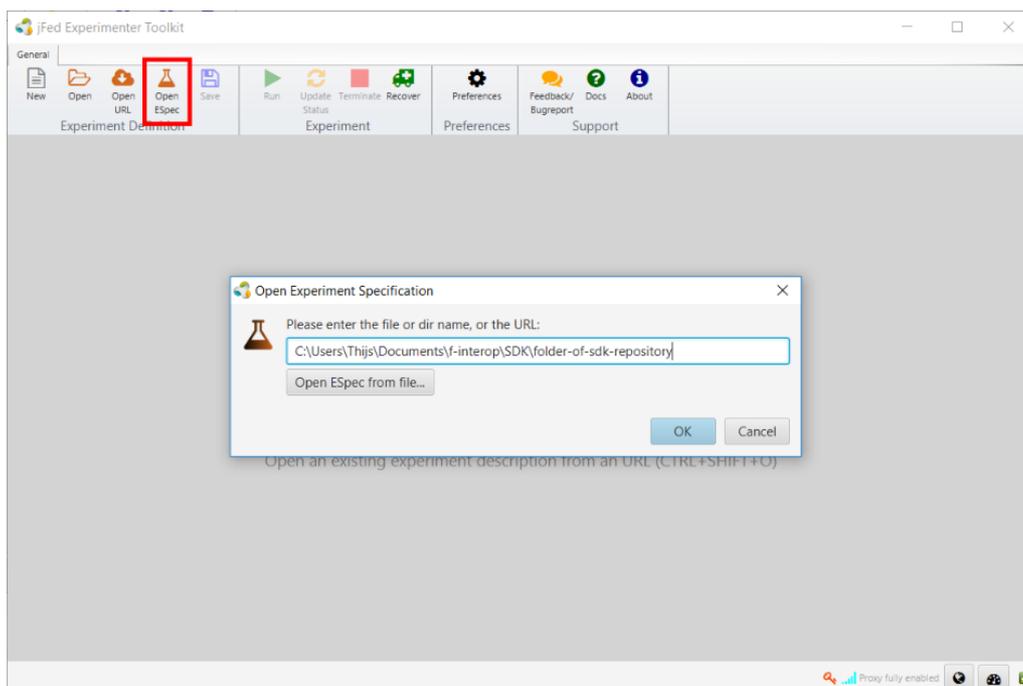


Figure 4: Pointing jFed to the base of the SDK/ESpec

4 Detailed architecture of a single test implementation

In the previous sections, the high level architecture was discussed. In this section we will have a closer look at the components and APIs involved in a single test. Figure 5 shows the involved components:

- Event Bus to exchange messages (AMQP based, we use RabbitMQ, <https://www.rabbitmq.com> as an implementation)
- Web interface for the GUI
- Orchestrator which starts and stops test sessions
- Test specific test and analysis tools
- Devices on testbeds or at the desk of the user, which communicate through an Agent with the event bus and the other components

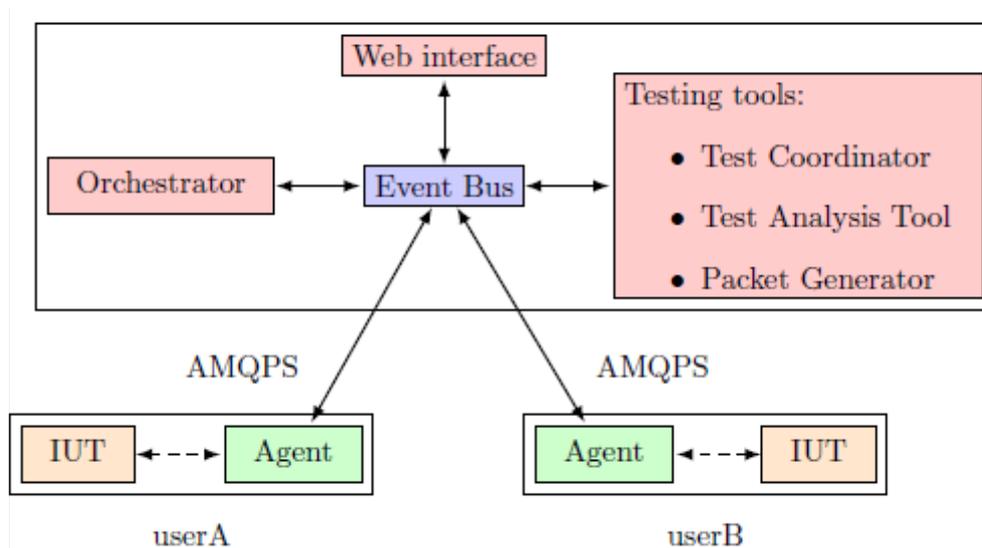


Figure 5: Detailed architecture of a single test

4.1 AMQP message bus

The message bus is used for two types of messages:

- Control plane messages relate to the management of an ongoing test session: e.g. start sniffer, signal the start/end of a test case, etc.
- Data plane messages contains the raw data exchanged between the implementations under test (IUT). The reason why it is possible to send data plane messages over the message bus, is that we send even link layer messages between two remote devices, or that we can use IPv6 even if the networks in between are not fully IPv6 enabled, and also that the messages are secured. It also allows to attach a sniffer during the test which can then send the data to an analyser.

JSON is the data format used for the application messages exchanged over the bus.

4.2 Test Coordinator

The **test coordinator** is the component that glues all parts together for a single test. Its purpose is to coordinate the test cases execution during a test session. It's a generic executor of the test case, although at the bootstrapping phase it retrieves a test session context from the orchestrator.

In the case of interoperability testing, it triggers actions in the following components:

- **SNIFFER** : This is the component handling the sniffing of the traffic exchanged between the participants of the test session
- **GUI**: Messages related to the execution (e.g. execute test case 2 , step 5)
- **AGENT**: configure interface, start interface, launch automated step
- **Test Analysis Tool (TAT)**: analyse test, e.g. captured traffic

This component is protocol agnostic, once instantiated it gets the specific context from the orchestrator and feeds from the Test Extended Descriptions (TED yaml files), see D2.1 for details. TEDs play a very important role in the architecture, they describe in detail each test case execution. TEDs are designed to be easy to write using YAML language, and they can also be written by the contributors.

4.3 Test Analysis Tool (TAT)

The TAT provides F-Interop with protocol analysis services. The TAT is the component that performs the verification of traces during a test session. F-Interop provides TATs for different protocols under test.

The TAT issues three types of verdicts:

- **PASS** when test purpose of the test case is verified
- **FAIL** when there is at least one fault
- **INCONCLUSIVE** when the behavior of the IUTs does not apply to the one described in the test purpose.

The architecture supports TATs which perform:

- step-by-step analysis
- analysis post mortem

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 that a TAT will receive from the AMQP Event Bus, and the format of the messages it will produce.

The TAT may implement its own dissector, and in that case it has to implement all the services described for the Dissector component as well.

4.4 Protocol Dissector

This component analyses the packets captured by the sniffer and dissects the packets into protocol specific packets (e.g. dropping non-relevant packets) and feeds them to the TAT.

4.5 Agent

The agent is a program a user downloads from the F-Interop website, and which allows him/her to connect an IUT to the F-Interop event bus. Communication between the agent and the event bus is authenticated and secured. Through the agent, the F-Interop platform can (remotely) interact with the IUT, for example by changing configuration or injecting packets. Similarly, the agent reports events to the event bus, such as sniffed packets.

4.6 Packet generator

In some conformance tests, a packet generator component can be used to generate and send packets towards the IUT. The component can for example implement the behaviour of a CoAP client when the IUT implements a CoAP server. A F-Interop client can purposely generate wrongly formatted messages to verify the correct behaviour of the IUT in such situations.

4.7 APIs in a single test

The communication (AMQP messages) between the components in this section is detailed at <http://doc.f-interop.eu>. This is called the “F-Interop API” in a single test tool.

5 Envisioned interaction for an end-user

An end-user who wants to do an F-Interop test (interop, conformance, performance, ...) has to go through the following F-Interop session steps (also detailed in D1.1). All interaction is done through the F-Interop GUI. The end-user doesn't need to know the internal details of the F-Interop APIs.

- User registration and login
- Registration of devices at the user's desk if any
- User starts by discovering the available test suites and by selecting the one he/she wants to execute.
- User specifies/selects resources in the F-Interop-Platform that are needed for his/her F-Interop session including the location models (see before), testing tools, libraries, etc. During this phase F-Interop-Platform may request information from the user or provide information to user for a coherent selection of the required resources.
- The resources selected in the previous step are actually reserved.
- The instantiation of the F-Interop-Platform resources that fit best with the user needs is done.
- In case of local resources, the user has to download and execute the Agent locally which needs to talk to the local device.
- The online F-Interop test campaign is launched and the selected test suites are executed against the IUTs.
- 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 upon request in case for example the user wants to apply for a certification/labelling program.
- 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.) has to remain accessible beyond the F-Interop session for the involved parties.

This user does not have to know the details on all components involved, all interaction is done through the F-Interop GUI website and the local Agent in case the user has local devices on his desk.

NOTE: for this initial version of the architecture (and accompanying implementation), we will focus on instant allocation and provisioning of the resources. Future reservation of resources will be for the next architecture.

6 Envisioned interaction for a contributor and SDK

In D1.1, an F-Interop-Contributor (FI-Contributor) was defined as any entity that provides testing tools or improves existing testing tools in the F-Interop-Platform as well as testbeds and devices that are added to extend the existing available resources.

Adding devices is as simple as described in the previous section where the interaction can be done through the web-based GUI and where a local Agent is needed to run near the devices.

However, for adding or improving tests (e.g. through the F-Interop open calls), the following interaction and workflow is needed:

- User registration and login
- User starts by discovering the available test suites and by selecting the one he/she wants to extend or use as template for a new test.
- The F-Interop platform deploys a full instance of the platform on a testbed (automatization of setup and configuration is done through Ansible Playbooks)
- User adapts/finetunes/debugs the test or defines a new test and can contribute back by giving the (adapted) code and Ansible Playbooks
- the F-Interop operators can verify the automatic deployment and test itself and make it available to others as well

All the information needed so that a contributor can make this kind of contributions to the test suite portfolio of F-Interop, is called the SDK (software development kit), and consists out of the following at this moment:

- this deliverable D1.4
- the documentation of the AMQP messages and single test framework at <http://doc.f-interop.eu>
- the full automatic deployment of the F-Interop software as described in this section. This was released end of April 2017 for milestone MS18 (Testbed as a service integration and SDK demonstrated) at month 18.

7 Conclusion

This deliverable is the final of two deliverables in the task 1.3 Architecture and FIRE+ integration design (D1.3 was the first iteration). The requirements studied in task 1.1 and 1.2 were used as input, together with feedback during the project of WP2, WP3 and WP4. We describe first the different location models (location of devices and users) that we will support in F-interop, followed by the current high level architecture of components in F-Interop and integration with FIRE+ testbeds. After that, we go into more details on the specific interaction between the components. This interaction is based on the AMQP message protocol and we have defined the message formats, functioning as an API when contributors want to add functionality to F-Interop.

We list also the envisioned interaction for an end-user and for a contributor. As such, the deliverable is also meant as part of the software development kit (SDK) towards (open-call) contributors.

8 Annex: Espec for sandbox

Below you can find the base file of the SDK/Espec for deploying a sandbox. The latest version can be found at <https://gitlab.f-interop.eu/f-interop-contributors/sdk> .

```
1 version: 1.0-basic
2 rspec: finterop.rspec
3
4 upload:
5   - bundled: session
6     nodes: [ server ]
7
8   - bundled: deployment/deploykey
9     nodes: [ gui ]
10
11 execute:
12   - bundled: enable-internet.sh
13     sudo: true
14   - direct: |
15       #!/bin/bash -x
16       cd session
17       chmod +x *.sh
18       sudo apt-get install -y jq
19
20     nodes: [ server ]
21
22 ansible:
23   host:
24     type: EXISTING
25     name: ansible
26     upload: deployment
27     execute:
28       - bundled: enable-internet.sh
29         sudo: true
30       - direct: |
31           #!/bin/bash
32           whoami
33
34           cd deployment
35           chmod +x install-ansible.sh
36           sudo ./install-ansible.sh
37
38           chmod +x get-files-and-deploy.sh
39           sudo ./get-files-and-deploy.sh
40
41   playbook:
42     - path: deployment/f-interop_ietf/ansible/deploy.yml
43       extra-vars: '@deployment/f-interop_ietf/ansible/finterop_vars.yml'
44     - path: deployment/gui/ansible/deploy.yml
```