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

Deliverable D2.3
Testing tools instantiations (initial report)

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:	**July 2017**
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	ETSI
Author(s)/editor(s)	Ghada Gharbi, Federico Sismondi, Remy Leone, Loïc Baron, Eunah Kim, Michael Hazan, Cédric Crettaz
Version	**1.0**
Keywords	Interoperability Testing, Conformance Testing, Remote Testing, Online, Platform, Testing components, Test enablers

Abstract

This report corresponds to the deliverable: **D2.3 – Testing tools instantiations – initial report.**

The deliverable D2.3 is the initial report of the F-Interop testing tools instantiations. It is the first version of how F-interop testing tools enablers described in deliverables D2.1 and D2.2 are instantiated to perform interoperability and conformance testing of CoAP and 6TiSCH protocols, and oneM2M standard for M2M communications. It describes the realized session models.

Table of Contents

List of Figures

List of Tables

List of Acronyms

ABC	Attribute Based Credential
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
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
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
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
R&D	Research & Development
SME	Small Medium Enterprise
SMS	Short Message Service
SOTA (or SoA)	State Of the Art
SSL	Secure Sockets Layer
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

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.

1.2.2.2 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.

1.2.2.3 T2.3: Testing tools instantiations M7-M34 (Task Leader: ETSI)

The purpose of this task is to verify the applicability of the tools developed in T2.1 and T2.2 to real standards. We therefore have identified the following emerging standards which we believe are of particular importance for the IoT: 6TiSCH, 6LoWPAN, CoAP, IPv6, oneM2M, Web of Things. For each of them, we will instantiate generic concepts developed in tasks T2.1 and T2.2. For each, we will comply with the following 2-step approach:

1. Develop the test specifications and conduct remote on-line testing in simple scenarios with a limited number of communicating devices.
2. Extend the testing to large scale remote on-line testing and experimentations with complex large scale scenarios. This will include adding the relevant test cases.

Roles: ETSI will lead the task, and will integrate contributions from Inria, UL and EANTC.
Outcomes: remote online testing tools and test suites for 6TiSCH, 6LoWPAN, CoAP, IPv6, oneM2M, Web of Things.

1.2.3 Deliverable Objectives and Methodology

1.2.3.1 Deliverable objective

The objective of the **D2.3 Testing tools instantiations – initial report** is to demonstrate the feasibility of the methods and adaptations developed in tasks T2.1 and T2.2 for online remote interoperability and conformance testing. The work to be done is to illustrate the instantiations of the F-Interop online remote interoperability-testing framework components while considering test suites for emerging IoT standards such as 6TiSCH, CoAP and oneM2M.

1.2.3.2 Deliverables methodology

We have considered the two types of testing tools in WP2 that the F-Interop should 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 protocols 6TiSCH and CoAP, as well as the standard oneM2M. For each of these two types of testing and these two selected protocols (6TiSCH and CoAP) and standard (oneM2M), 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 start 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, CoAP and oneM2M, 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.

Table 1: F-Interop Session

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 setup	The instantiation of the F-Interop-Platform resources that fits 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. he test results and verdicts are provided together with explanations in case of FAIL or

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

		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.

The work reported in this document, that corresponds to the deliverable D2.3, covers each step of a F-Interop session.

Internal discussions with partners involved in WP2 as well as with the whole consortium helped identify 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 following sections describe the architecture of the testing tools, its components and other auxiliary components for performing interoperability and conformance tests.

2 F-Interop platform enablers

In this section, the F-Interop testing architecture is presented in Figure 1. The components of this architecture are responsible of managing the testing infrastructure, 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.

In this section, the F-Interop testing architecture is presented in Figure 1. The required core enablers are: (i) the event bus responsible of exchanging messages between F-Interop platform, (ii) session orchestrator which plays an administrative role, (iii) resource repository containing a list of all devices and their properties that can be used for F-interop tests, (iv) result repository to store test results, (v) GUI the first point of contact for an end-user, and (vi) agent running near devices (e.g. at the user premises) to communicate with the orchestration and analysis components. The interaction between components is based on the AMQP message protocol and an API was elaborated to define AMQP message formats.

Through standard security mechanisms, the architecture ensures the authentication of different users, and the confidentiality of test results.

2.1 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.

This architecture enables may independent components to talk to each other and avoid a monolith that would be difficult to manage as the number of components increase. Each message contains a routing key which indicates how to route this message to the relevant input queues of the components. It also shipped with a web interface for monitoring the load and enable easy debugging. Isolation of test session is performed using virtual hosts. The exchanged messages and the routing keys are standardized by F-Interop, following an API.

The messaging patterns is used by:

- Components which can publish events (the component is a producer): commonly used for send debug logs, send data to other IUTs or controlling the execution of a test
- Components can subscribe to events: (the component is a consumer): e.g. GUI reads test informative messages and prints them in the graphical interface.
- Components can provide or call services (components are consumer/producer at the same time for this type of interaction): basically, we have two types request/replies or action/confirmation and they consume certain messages, and reply by producing others as a result.

The communication between F-Interop testing enablers is detailed at http://doc.f-interop.eu/.

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.

We notice that in the case of performance testing, only control traffic is managed. In this deliverable, only interoperability and conformance testing tools are considered.

An F-Interop-User conducts remote interoperability tests in independent sessions. We use the virtual host mechanism of RabbitMQ [2] 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.

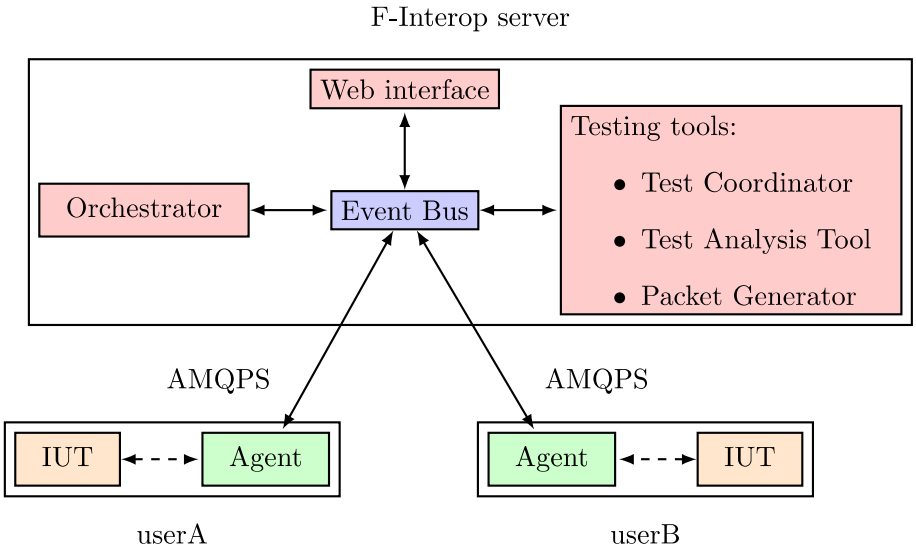

Figure 1: F-Interop remote interoperability testing architecture

2.2 Components description

2.2.1 Agent: connecting F-Interop users to the F-Interop Platform

An **agent** is a program that an 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 an administrative role. The orchestrator main roles are:

- monitoring the users that are connected,
- activating the rooms currently in use and starts/stops the test sessions. It is also in charge of provisioning the message broker and updating Technical Overview of F-Interop firewall rules when test sessions are activated. It does so by spawning/killing the processes of the different components connected to the event bus. It uses the supervisor [3] process control system.
- publishing signalisation events to the event bus:
    - configuration message of the session to the event bus on session start,
    - terminate message on session close.

The test orchestrator can be run as a central service for a specific F-Interop test (think a user just running a test), or can be deployed on a testbed virtual machine (think a user wanting to adapt a test or develop a new test).

The orchestrator relies on RESTful API to communicate with F-Interop components. Possible operations are:

- Creating a new session,
- Starting a session,
- Stopping a session,
- Deleting a session,
- Creating a new user,
- Getting information about a user,
- Deleting a user,
- Posting an event on the event bus.

The orchestrator exposed RESTful API is detailed at http://doc.f-interop.eu/#session-orchestrator.

2.2.3 Testing tool

The **testing tool** handles the components throughout an F-Interop session. Each testing tool test either a particular protocol (e.g. CoAP Testing Tool) or a set of protocols used in a certain architecture (OneM2M Testing Tool). 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 perform the tests between IUT(s) and generate a verdict for each test case. It operates as a black box which emits coordination messages towards agents (accurately in case of interoperability and conformance testing), and generates test verdicts and a final report. For interoperability and conformance testing, F-Interop provides a reference implementation for testing tools and documents the format of the messages exchanged between components (see doc.f-interop.eu). This description of messages exchanged between internal components of the testing tool can be seen as a second level (or extension) of the previously mentioned API specification. The purpose of this is to ease development and integration of new testing tools in F-Interop environment. This reference testing tool architecture and implementation includes components such as a test coordinator, a test analysis tool, a sniffer, a

packet router (which intentionally can drop packets for simulating lossy links) and a packet generator (mainly for conformance testing). You can find more details in deliverable **D1.3_v1_0: Initial architecture design** and in doc.f-interop.eu. This reference testing tool architecture is the one used for CoAP interoperability testing tool, and OneM2M interoperability testing tool.

In order to support a collaborative effort and a coordinated code development and integration of the testing tools, the code of each testing tool is shared and made available through the dedicated GitLab. GitLab permits not only to store the source code using a performant versioning system, but also to provide continuous integration and continuous deployment to each F-Interop developer. This facilitates the control of the versions and ensures that the F-Interop partners are running the very last version of each testing tool. The other components of the F-Interop architecture are also saved into this GitLab. Our GitLab is composed in two groups: the first group contains the private source code of the F-Interop core platform and the second group includes the public source code related to the open call and the associated SDK. Of course, the GitLab repository offers an issue tracking feature which lists the different bugs or problems encountered with the current version of the F-Interop platform.

2.2.4 GUI

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.

2.2.5 Resources Repository

The resources repository provides a pivotal function to enable the various testing tool to share a common registry and set of identifiers for each IUT. The resource repository includes a database listing all the resources and testing tools available in the F-Interop platform, including the testing tools. As a part of the F-Interop central services, the resources repository is connected to the other platform components through the AMQP bus. The resources repository is continuously listening to the AMQP bus, waiting on the requests from the other components like the different testing tools and the graphical user interface (GUI). The resources repository offers the possibility to the other components to add a new resource, to update it, to get it and finally, to delete it. The API provided by the resources repository is described here: http://doc.f-interop.eu/#resource-repository-rr. Different kinds of resources are stored inside the resources repository: IoT devices (sensors or actuators) installed in the different testbeds, virtual machines, testing tools, IoT devices under test provided the end-user, etc. In summary, the resources repository is a map of the current status of the F-Interop platform by listing the available resources connected to the testing tools.

3 Status of work

3.1 Status of work for T2.3 – Deliverable D2.3

The work accomplished for the first iteration of the task T2.3 - Testing tools instantiations - is to verify the applicability of the tools developed in T2.1 and T2.2 to real standards. Following the F-Interop architecture, the defined interfaces and the messaging API, interoperability, conformance, and performance testing tools are developed by the F-Interop partners. This document focuses on work done in interoperability and conformance testing. Emerging IoT Standards such as 6TiSCH, CoAP, and oneM2M were studied to illustrate the instantiations of the F-Interop remote testing framework components.

CoAP testing tool had reached its beta version. It is used as the reference implementation for testing tool testing protocols which sit top of IP. Its architecture and protocol agnostic implementation is documented and provided for other partners and contributors in the form of an enabler for interoperability testing. Integration of CoAP testing tool to the other F-Interop enablers and ecosystem (GUI, resource repo, results repo, orchestrator, etc) has been achieved.

6TiSCH and oneM2M testing tools are at early stage. Details about 6TiSCH, CoAP and oneM2M testing tools are given in the following sections.

3.2 CoAP interoperability testing tool

3.2.1 Brief overview

The CoAP interoperability testing tool follows the reference testing tool architecture and API defined by F-Interop. As described previously in this document (see Testing tool section) and other deliverables and documents (doc.f-interop.eu, D2.1 and D1.3), the reference interoperability testing tool architecture comprises:

- Test Coordinator: component which coordinates the test cases execution during a F-Interop session. Emits the verdicts at the end of each test case, and a report at the end of the session.

- Test Extended Description: set of documents in yaml format which describes in detail each steps and actions of a test case execution.

- Test Analysis Tool: component which 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.

- Packet dissector: Component for decoding network traces provided as a pcap file, and which returns human readable description of them including headers, fields, values etc.

- Packet sniffer: component which handles the sniffing of the traffic exchanged between the participants of the test session.

- Packet router: component which forwards data plane messages between IUTs and which can intentionally drop packets for simulating lossy links.

- Packet Generator: component for the generation and the dispatch of packets towards the client or server endpoints

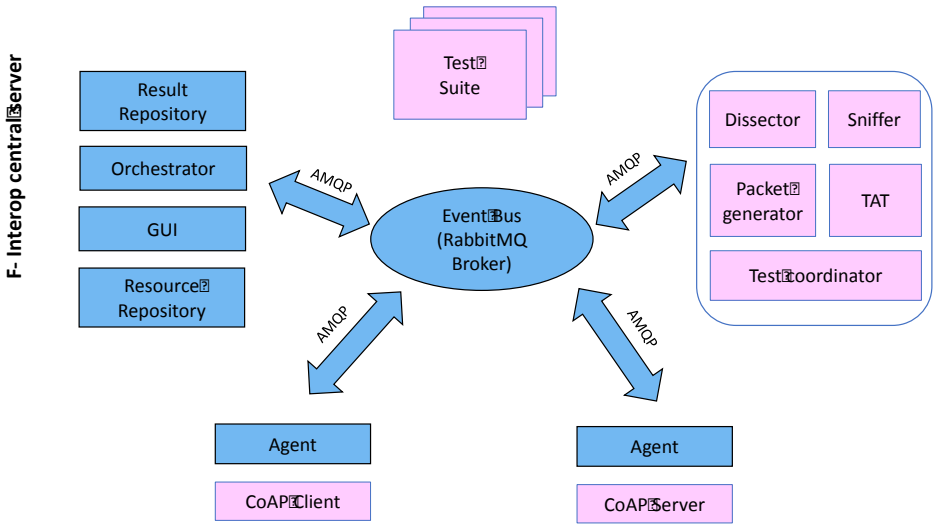

Figure 2: CoAP interoperability testing tool architecture

3.2.2 Session models

The CoAP interoperability testing tool currently supports two of the main session models targeted by F-Interop:

- Remotely located user-assisted IUT vs user-assisted IUT (see Figure 3). The user-assisted (passive) testing approach minimizes the control of the IUTs and bases the emitted verdicts on the observed interactions.

- Remotely located user-assisted IUT vs automated IUT (see Figure 4). The automated (active) testing approach is defined when test system controls all the actions to be done by the IUTs by alternating stimuli and observations.

By remotely located we mean that both IUTs are run in two geographically different locations. For these use cases, the CoAP packets exchanged between IUTs are sent, forwarded and received using the AMQP event bus's data plane, and using the agent as a proxy.

The "remote user-assisted IUT vs user-assisted IUT" is one of the main targeted session models of F-Interop as the setup is akin to the setup used in face-to-face interoperability events.

The following Figure describes the setup and interaction between IUTs and F-Interop's core enablers for interoperability testing.

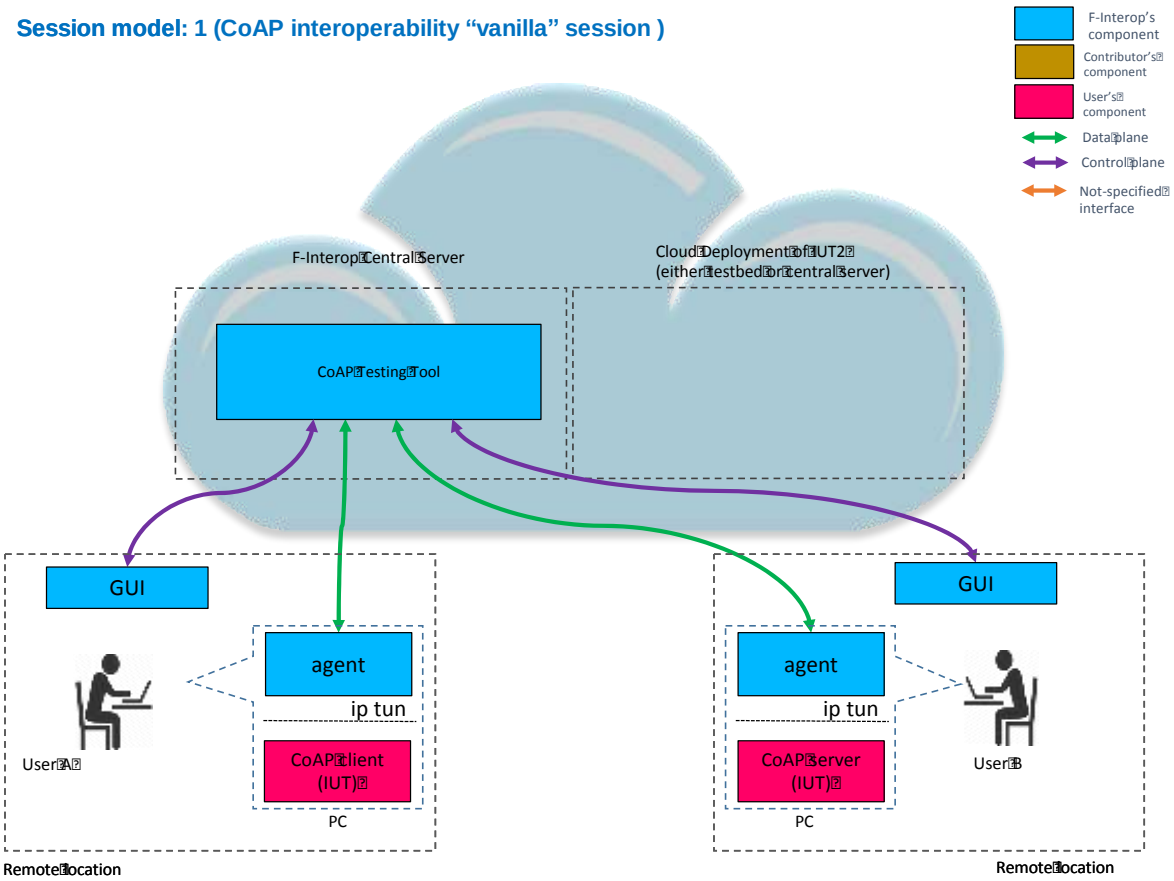

Figure 3 - CoAP interoperability testing session model: two remote user-assisted IUTs

Also, with purpose of enabling the "test-on-demand" feature of F-Interop online platform, CoAP interop testing tool can be instantiated with one user and one automated-IUT. **Automated-IUTs** are implementations under test which are fully automated and are integrated into F-Interop in such a way that they can be driven by F-Interop though the execution (specifically by the test coordinator component of the testing tool). This means that a user doesn't need to schedule a session with another user to run an interoperability session, as automated-IUTs are always online and ready to be run.

The following diagram describes the setup and interaction between IUTs, automation components and F-Interop's core enablers for interoperability testing.

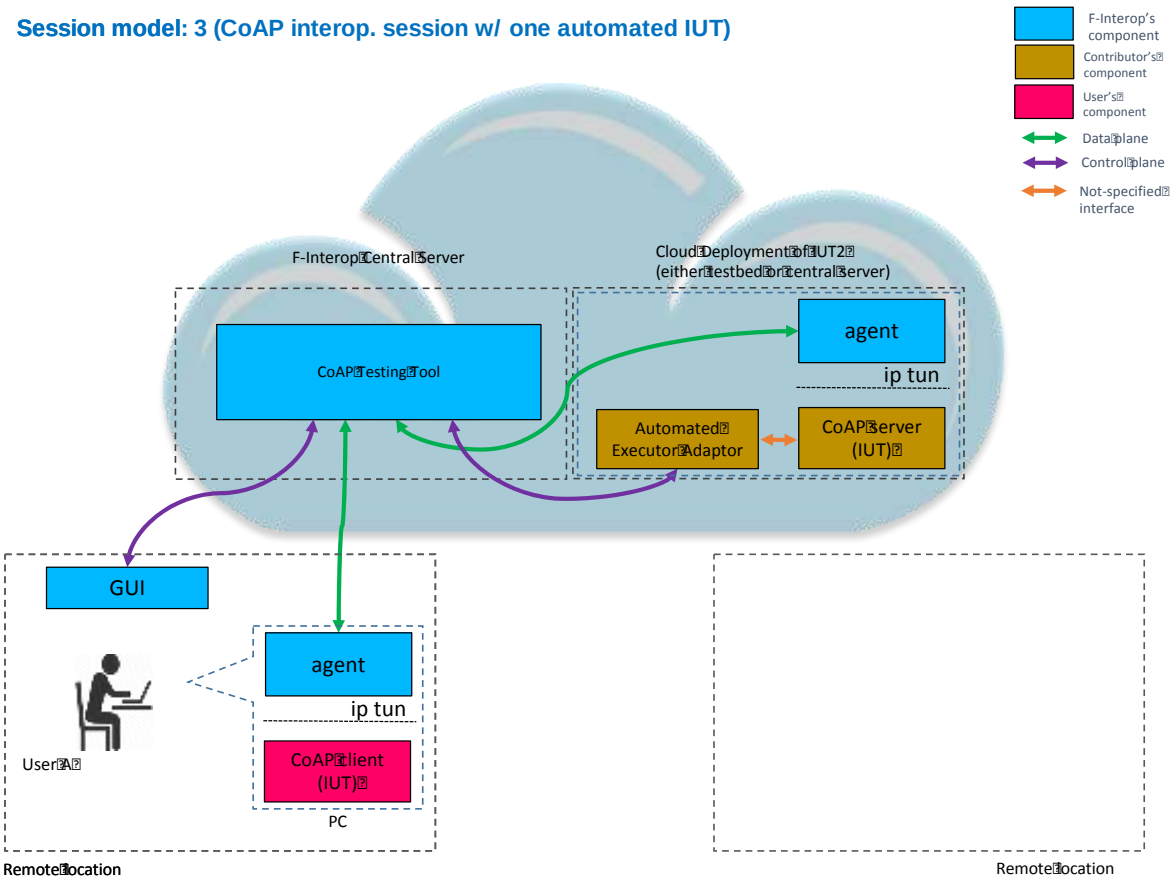

Figure 4 - CoAP interoperability testing session model: user-assisted-IUT vs automated-IUT

3.3 6TiSCH conformance testing tool

The 6TiSCH protocol aims to bring IPv6 connectivity to wireless sensor networks by using high-reliability time-slotted communications (TSCH). The reference implementation of 6TiSCH is OpenWSN, a BSD-licenced operating system for constrained devices. The IETF 6TiSCH working group regularly organizes plugtests to improve the state of interoperability between different 6TiSCH implementation. Therefore, they have a great interest in having a testing tool that support remote testing.

We worked with this group to set up a series of Test Descriptions that will be supported by the 6TiSCH testing tool. Those test descriptions are updated version from previous test descriptions used in previous interoperability plugfest.

The latest version of those test description (https://bitbucket.org/6tisch/td-6tisch/src) was published on the 6TiSCH mailing list and will be used at the F-Interop 6TiSCH plugtests.

Interoperability Test Description			
Identifier:	TD_6TiSCH_6P_01		
Objective:	check a 6P COUNT command is successful		
Configuration:	• topology => a DAGroot and a 6N within range (single hop) • all devices configured to communicate on a single frequency (frequency to be allocated during the event) • link-layer security disabled		
References:	• IEEE802.15.4-2015 • RFC8180 • draft-ietf-6tisch-6top-protocol-05		
Pre-test conditions:	DAGroot and 6N are turned off		
Test Sequence:	Step	Type	Description
	0	Stimulus	• turn on the DAGroot • turn on the 6N
	1	Check	the 6N synchronizes to the DAGroot
	2	Stimulus	• have the 6N send a 6P ADD request to the DAGroot - asking for a single TX cells from 6N to DAGroot - with the candidate cellist set to [(4,5)] • how this is done is implementation-specific
	3	Check	the 6P transaction is succesful and both the DAGroot and 6N have added a cell from 6N to DAGroot at slotoffset=4 and channeloffset=5
	4	Stimulus	• have the 6N send a 6P COUNT request to the DAGroot • how this is done is implementation-specific
	5	Check	check that the sniffer captures this 6P transaction
	6	Check	check that the format of the 6P request and response is according to draft-ietf-6tisch-6top-protocol-05
	7	Check	• check that, in the 6P response, the returned code SUCCESS • check that, in the 6P response, the count value is 2

We are organizing, through the LIST open-call project, the first F-Interop 6TiSCH plugtest. It will take place in Prague July 13th and 14th prior to the IETF meeting. (http://www.etsi.org/about/10-news-events/events/1197-6tisch-interop-prague-2017)

3.3.1 Brief overview

The 6TiSCH testing tool can be launched from the F-Interop platform from the GUI or from an API. When launched, it communicates with the agent to sniff packets and perform testing on them. Once the testing is performed, a verdict is sent on the AMQP bus so that all other components can be notified and move on with the testing workflow.

At its core, the 6TiSCH testing tool uses Wireshark to dissect all packets. The architecture of the testing tool is agnostic to the protocol it aims to dissect. There are plans to develop a set of tools to use this architecture to quickly bootstrap a new testing tool of a protocol supported by Wireshark. Because Wireshark supports a large number of network protocols, this new set of tools will allow the F-Interop project to quickly adapt to new protocols and bootstrap new testing tools.

3.3.2 Session models

Because of the time-sensitive requirements of synchronization used in 6TiSCH, all testing is happening locally. All nodes need to be deployed in the same location for a 6TiSCH network to be set up. Because of that, at the moment only local testing is supported. But once the testbeds are made available in the F-Interop platform, remote participants will be able to deploy their implementations to a testbed and test it there.

3.4 oneM2M interoperability testing tool

3.4.1 Brief overview

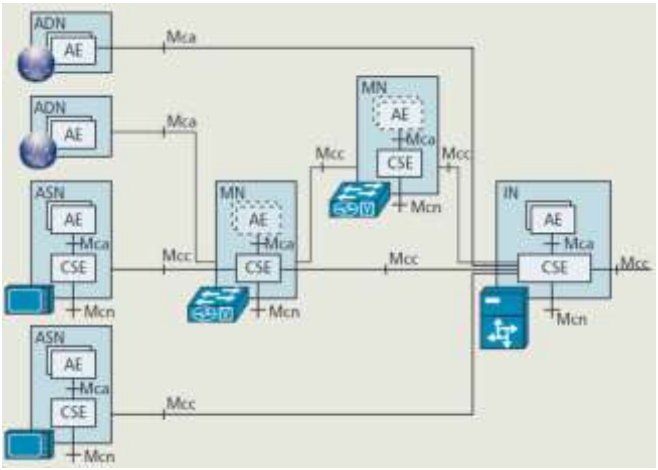

Figure 5: oneM2M functional architecture

The oneM2M global initiative [4] is an international partnership project established in June 2009 by the seven most important SDOs in the world and various alliances and industries. The main goal is to define a globally agreed M2M service platform by consolidating currently isolated M2M service layer standards activities.

The oneM2M standard is organized into five technical working groups focusing on M2M requirements, system architecture, protocols, security, and management, abstraction and semantics.

As described in Figure 5, the oneM2M system architecture is composed of the following four functional entities: the application dedicated node (ADN); the application service node (ASN); the middle node (MN); and the infrastructure node (IN). Each node contains a common services entity (CSE), an application entity (AE), or both. An AE provides application logic, such as remote power monitoring, for end-to-end M2M solutions. A CSE comprises a set of service functions called common services functions (CSFs) that can be used by applications and other CSEs. CSFs include registration, security, application, service, data and device management, etc.

The oneM2M standard adopted a RESTful architecture, thus all services are represented as resources to provide the defined functions.

3.4.2 OneM2M interoperability testing tool design

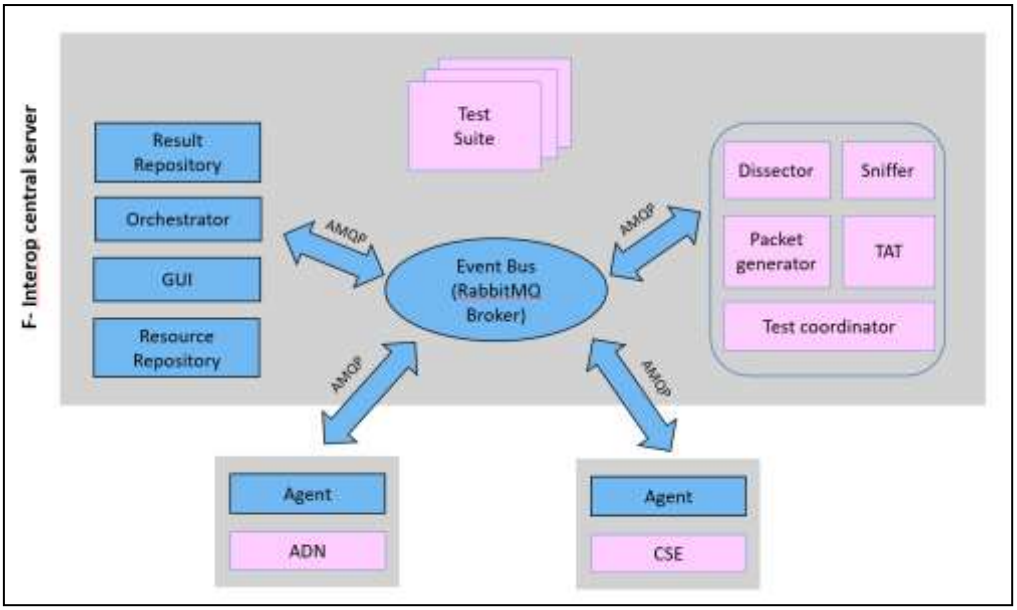

Figure 6: oneM2M interoperability testing tool architecture

As described in Figure 3, the oneM2M interoperability testing tool includes the following specific components:

- The oneM2M testing tool which is composed of the subsequent modules:
  - The Dissector for decoding network traces captured by the sniffer.
  - The Sniffer for sniffing the traffic between the oneM2M interoperability testing components during a session.
  - The packet generator for the generation and the dispatch of packets towards the client or server endpoints.
  - The Test Analysis Tool (TAT) for the verification of traces of the test session. The verification is post mortem.
  - The Test coordinator responsible of the management of oneM2M test suites achievement during oneM2M session.

  All the communication between the components respect the API available at http://doc.f-interop.eu/ .
- A oneM2M Agent to connect oneM2M endpoints (ADN and CSE implementations) to the AMPQ broker to enable the interaction with F-interop central server.

These components interact with the F-Interop generic services such as the Resource Repository, the Orchestrator, the GUI, and the Resource repository to conduct remote interoperability testing.

3.4.3 Session models

We target to implement the following session models.

As depicted in Figure 7, a oneM2M interoperability session will be established between an Application Dedicated Node (ADN) which plays a client role and a

Commons Services Entity (CSE) implementation which plays the role of a server. These implementations are provided by F-INTEROP users.

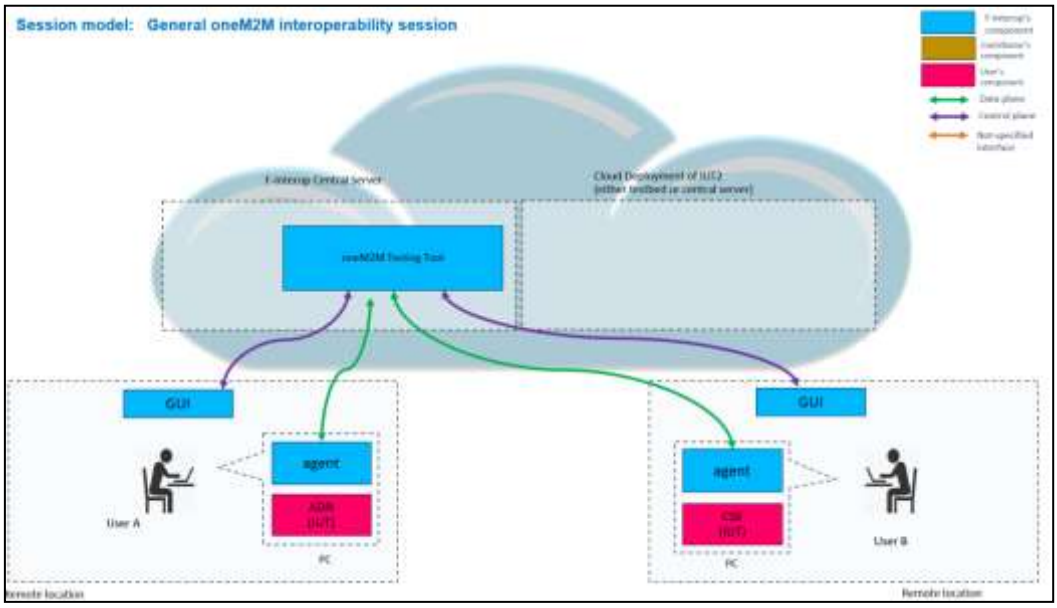

Figure 7: oneM2M interoperability testing - session model 1

The Figures 8 and 9 describe two ways to enable reference-based oneM2M interoperability testing. In the first place, a AND client is considered as target IUT while a CSE implementation plays the role of reference-based interoperability testing. This case is illustrated in Figure 5. In the second place, a CSE server is considered as target IUT while an AND client plays the role of a reference-based interoperability testing. This case is depicted in Figure 6. The ADN and CSE reference-based interoperability testing implementations will be provided by F-Interop contributor (ETSI).

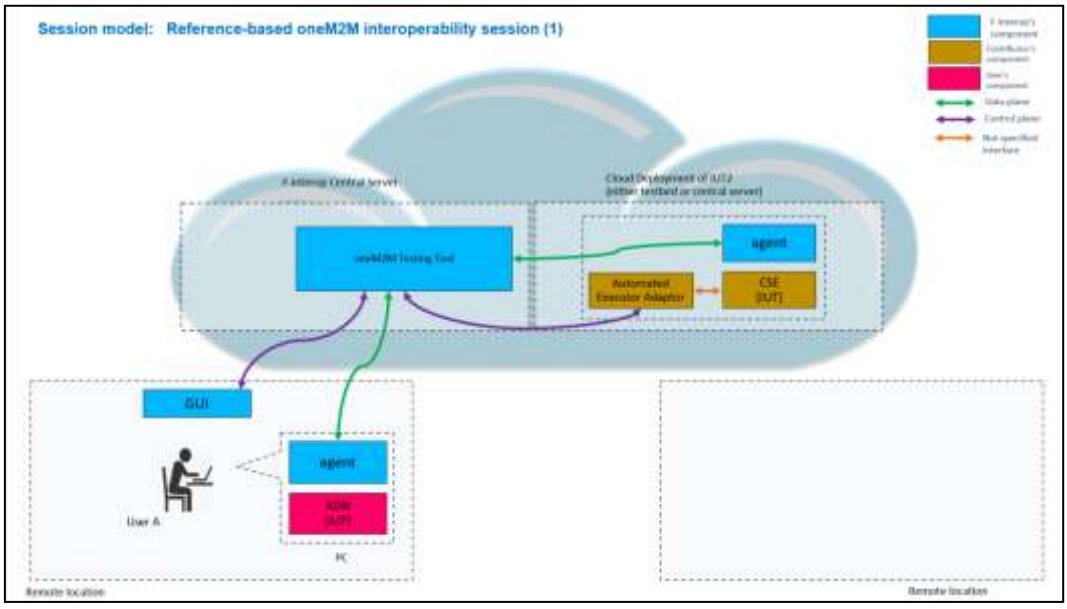

Figure 8: oneM2M reference based interoperability testing - session model 2

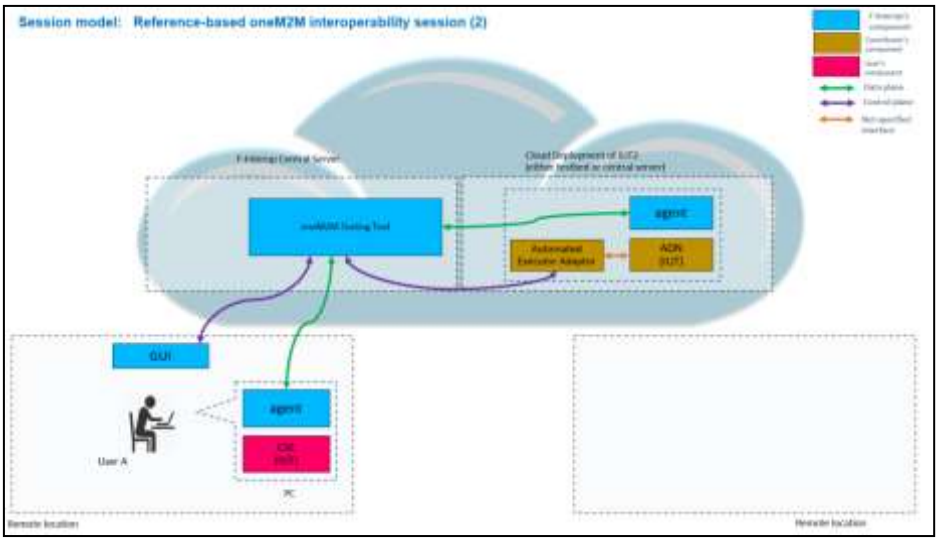

Figure 9: oneM2M reference based interoperability testing - session model 3

4 Conclusion

This document presented the work related to task T2.3. We present indicators on the status of the work of the first iteration of interoperability/conformance testing tools instantiations. We first introduce the F-Interop framework core enablers that are responsible for managing the testing infrastructure, including provisioning the underlying network, capturing trace, starting/stopping the different tests, and reporting verdicts. Then, we describe the outcome of the work achieved in the development of these components based on IoT emerging standards (6TiSCH, CoAP, and oneM2M) interoperability/ conformance test suites experiences. It includes the development of the following components: The Agent, testing tool, Test coordinator, Packet generator, Test analysis tool). GUI, Resource Repository, Result Repository, and Session Orchestrator were designed and implemented to be re-used when implementing testing tools for various protocols.

The work to be done in next steps include:

- Putting the CoAP testing tool into production (as a beta version) for a user to user interoperability testing and also user against reference implementation (~conformance testing).
- Development of new additives test for other RFC rather than CoAP CORE (insert ref) and larger scale scenarios are foreseen for the period to come.
- Development of 6TiSCH testing tool (Dissection analysis, bootstrap of the F-Interop agent on embedded devices, …Development of the components of the oneM2M remote interoperability/ conformance testing tools.

5 References

[1] http://zeromq.org

[2] https://www.rabbitmq.com/

[3] http://supervisord.org/

[4] J. Swetina et al., “Toward a Standardized Common m2m Service Layer Platform: Introduction to onem2m,” IEEE Wireless Commun., 2014, vol. 21, no 3, pp. 20–26.

6 Annex

6.1 An example of oneM2M interoperability test description from oneM2M technical specification TS-0013-interoperability testing V2.2.

Interoperability Test Description			
Identifier:			TD_M2M_NH_06
Objective:			AE registers to its registrar CSE via an AE Create Request
Configuration:			M2M_CFG_01
References:			TS-0001 [1], clause 10.2.1.1 TS-0004 [2], clause 7.3.5.2.1
Pre-test conditions:			• CSEBase resource has been created in CSE with name {CSEBaseName} • AE does not have an AE-ID, i.e. it registers from scratch
Test Sequence			
Step	**RP**	**Type**	**Description**
1		Stimulus	AE is requested to send a AE Create request to register to the Registrar CSE
2	Mca	PRO Check Primitive	• op = 1 (Create) • to = {CSEBaseName} • fr = AE-ID • rqi = (token-string) • ty = 2 (AE) • pc = Serialized representation of <AE> resource
3		IOP Check	Check if possible that the <AE> resource is created in registrar CSE.
4	Mca	PRO Check Primitive	• rsc = 2001 (CREATED) • rqi = (token-string) same as received in request message • pc = Serialized representation of <AE> resource
5		IOP Check	AE indicates successful operation
IOP Verdict			
PRO Verdict			

6.2 An example of oneM2M interoperability Test Extended Description (YAML file)

```
---
testcase_id: TD_M2M_NH_06
uri: to add
configuration: M2M_CFG_01 (The AE manages resources on the registrar CSE)
objective: AE registers to its registrar CSE via an AE Create Request
preconditions:
   - CSEBase resource has been created in CSE with name {CSEBaseName}
   - AE does not have an AE-ID, i.e. it registers from scratch
references:
   - oneM2M TS-0001 clause 10.2.1.1
   - oneM2M TS-0004 clause 7.3.5.2.1
sequence:
   - step_id: TD_M2M_NH_06_v01_step_01
      - type: stimuli
      - iut: ADN
      - description: AE is requested to send a AE Create request to register to the Registrar CSE

   - step_id: TD_M2M_NH_06_v01_step_02
      - type: PRO check
      - description:
         - the request sent by the client contains
         - op = 1 (operation type)
         - to = {CSEBaseName}
         - fr = AE-ID
         - rqi = (token-string)
         - ty = 2 (AE)
         - pc = Serialized representation of <AE> resource (json)

   - step_id: TD_M2M_NH_06_v01_step_03
      - type: IOP Check
      - description: Check if possible that the <AE> resource is created in registrar CSE.

   - step_id: TD_M2M_NH_06_v01_step_04
      - type: PRO Check
      - description:
         - CSE send a response containing
         - rsc = 2001 (CREATED)
         - rqi = (token-string) same as received in request message
         - pc = Serialized representation of <AE> resource (json)

   - step_id: TD_M2M_NH_06_v01_step_05
      - type: IOP Check
      - description: AE indicates successful operation
...
```