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

Deliverable D4.1
Open API and Resource Repository

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:	**30.04.2017**

Dissemination Level		
PU	Public	X
CO	Confidential, only for members of the consortium (including the Commission Services)	

Document properties

Responsible partner	Device Gateway SA (DG)
Author(s)/editor(s)	Eunah Kim (DG), Matteo Fillipponi (DG), Cédric Crettaz (MI)
Version	1
Keywords	Resource repository, F-Interop, interoperability test, performance test, federated networks

Abstract

D4.1 is a demonstrator, and this document is voluntarily made to provide the overview of the demonstrator including the open APIs and the operation of the F-Interop Resource Repository. It includes architectural view of the resource repository showing the relationship with other components. It investigates the required functions for the resource repository by checking each step of F-Interop sessions in a viewpoint of FI-Users. It also shows the implementation of APIs and the examples of the operations.

The Resource Repository has been designed aligned with F-Interop Architecture described in D1.3, and carefully followed the requirements and session steps that described in D1.1. It collected all requirements from WP2 and WP3 for the specification of the resource information. The implementation and its integrated tests have been performed with T4.2 and T4.3. The implemented RR has been tested in several iterations integrated with other core components such as TBaaS and F-Interop GUI, and verified its functions and behaviours. Although the Task duration is until M18, it will be continuously maintained and updated during the remaining project lifetime as needed.

Table of Contents

List of Figures

List of Tables

List of Acronyms

AMQP	Advanced Message Queuing Protocol
API	Application Program Interface
CoAP	Constrained Application Protocol
ETSI	European Telecommunications Standards Institute
IETF	Internet Engineering Task Force
IoT	Internet of Things
IP	Internet Protocol
M2M	Machine to Machine
RFC	Request For Comments
RR	Resource Repository

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

WP4 is for research, development and integration of the F-Interop Testbed as a Service. It develops and demonstrates the open API and resources repository, develops the required SDK for third parties' developments and extensions and develops and optimizes the user interfaces.

1.2.2 Task Objectives

This task will research and develop the required API and resource repository to enable the integration and homogenization of the various branches of the F-Interop testbed as a service, including the devices under test, as well as the devices used in the three testbed federations. The task will take into account emerging RESTful standards as well as Fed4FIRE APIs, such as OML, FRCP, AM and Rspec. Based on this comparative analysis, the task will select and specify an optimal data format to be used by F-Interop common resource repository. Once the Open API defined, the task will work on the implementation of a highly scalable resource repository. The resource repository will be tested and improved through several iterations.

1.2.3 Deliverable Objectives and Methodology

This deliverable is to demonstrate the open API and F-Interop resource repository. Although D4.1 is a demonstrator, not a report deliverable, this document is voluntarily made to provide overview of the demonstrator. It includes architectural view of the resource repository showing the relationship with other components. It investigates the required functions for the resource repository by checking each step of F-Interop sessions in a viewpoint of FI-Users. It also shows the implementation of APIs and the examples of the operations.

2 Design of F-Interop Resource Repository

The F-Interop Resource Repository (RR) has been designed to provide and manage the resources for F-Interop interoperability and performance tests. The following sections show the architectural view of RR and the required functions in a viewpoint of FI-User. The F-Interop architecture and the required functions from the FI-Users have been carefully studied in designing of F-Interop RR.

2.1 Architectural view of F-Interop Resource Repository

The following Figures (Figure 1 and Figure 2) are copied from F-Interop D1.3 to discuss the functions of F-Interop RR in F-Interop platform. Figure 1 indicates the high-level architecture without involvement of the F-Interop federated testbeds. As it shows, the RR communicates with GUI when FI-Users want to obtain resources for a test. RR is in "F-Interop central services" and communicates with GUI via AMQP event bus same as the other F-Interop components.

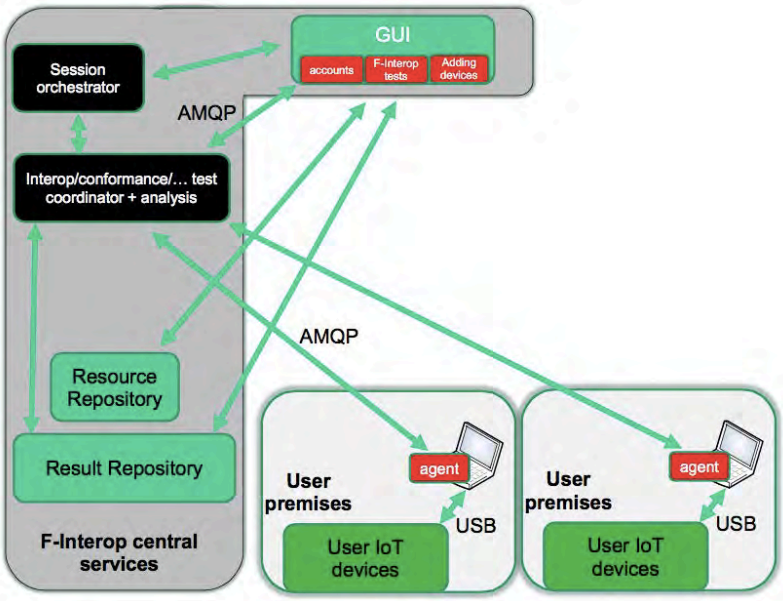

Figure 1 - F-Interop high-level architecture without testbeds (*source: D1.3*)

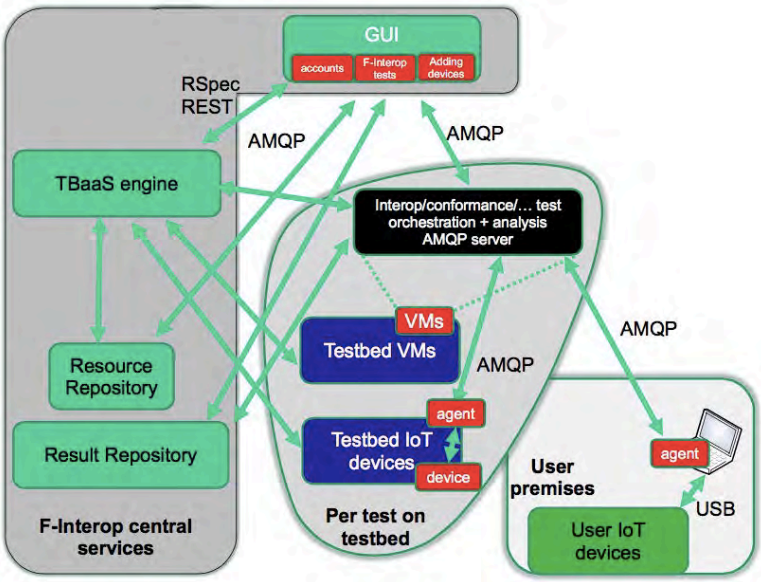

Figure 2 - F-Interop high-level architecture with testbeds (*source: D1.3*)

Figure 2 illustrates the high-level architecture with involvement of F-Interop federated testbeds. In this case, RR communicates with TBaaS for collecting and using testbeds resources. The TBaaS provides resource API with testbeds for collecting testbeds resources and sends F-Interop Resource Repository the information via F-Interop Eventbus. As TBaaS pushes the testbeds resources to RR, the RR maintains both resources from FI-Users or FI-Contributors and from the federated testbeds. As the two Figures depict only F-Interop components, the Figure 3 can help to see the interaction with the federated testbeds. As it shows every communication among F-Interop components is through F-Interop AMQP Eventbus. TBaaS takes care of federation with the testbeds including the communication for the testbed resources. It sends RR the information via the AMQP Eventbus. The RR receives the resource information and manages it to fit into the F-Interop operations.

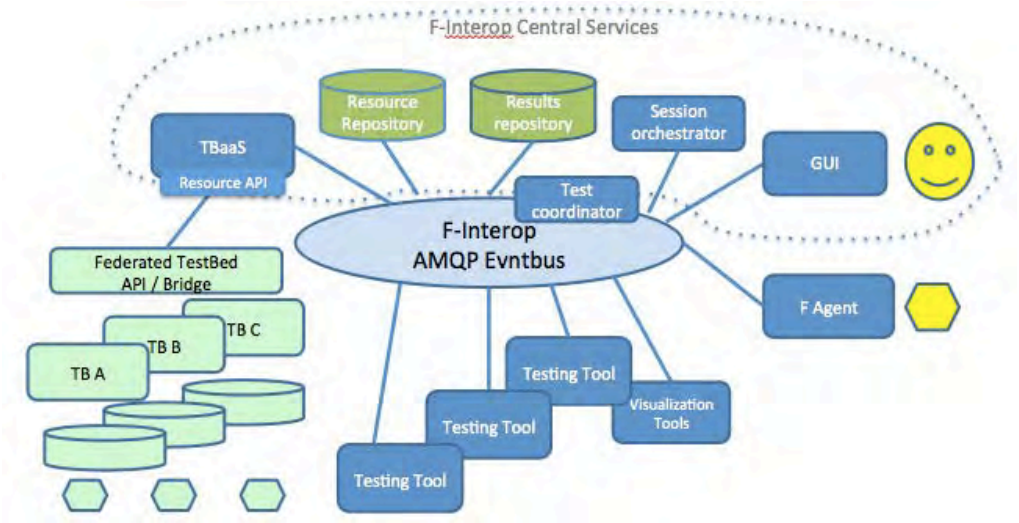

Figure 3 – F-Interop Eventbus diagram including Testbeds

2.2 F-Interop session and the role of RR

F-Interop D1.1 identifies the set of actions that an F-Interop User has to perform in order to remotely execute interoperability and performance tests using the F-Interop platform as explained in Table 1. The FI-User performs a set of sequential actions to test an interoperability or performance tests. The blue boxes show the actions that resource repository should provide. It is not described in the table, but resource release should be performed after the testing session ends.

In the Step 0, an FI-User gets authenticated and authorized in F-Interop Portal. Then the user registers IUT for its test as illustrated in Figure 4.

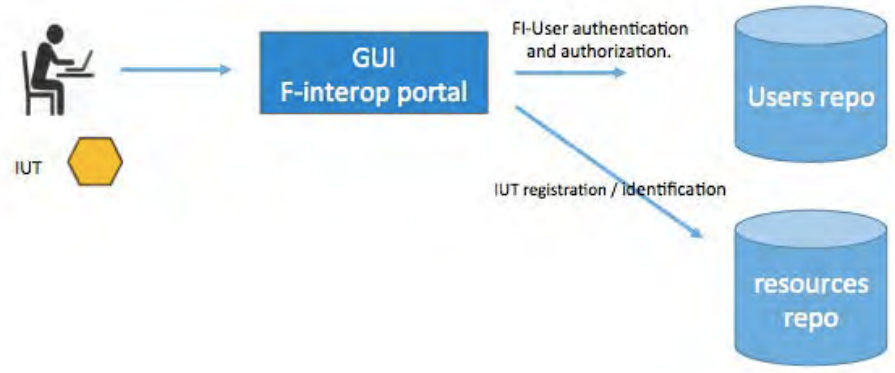

Figure 4 - FI-User Session: Step 0

Table 1. F-Interop Session in a viewpoint of FI-Users (*Source: F-Interop D1.1*)

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 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 analyzed. 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.

The FI-User searches available testing suites and resources for the desiring test through Step 1 and 2 as shown in Figure 5. In this step, after selecting a suitable testing suite, FI-Users indicate the required resources for finding suitable resources for the planned test. Some examples are as followings:

- show me all 802.15.4 compatible nodes in Switzerland.
- show me all nodes with OpenMoteSTM and TelosB.
- show me all available resources for OneM2M test.
- show me all nodes supporting CoAP.

The FI-Users express such needs through F-Interop GUI. When RR receives a request from GUI, RR returns the available resources and the corresponding detail information of the resources as illustrated in Figure 5. It is noted that this Figure does not include the interaction with TBaaS for collecting testbeds resources. As it is explained in the Section 2.1, TBaaS sends the testbeds resources to RR and the RR maintains both resources from FI-Users or FI-Contributors and from the federated testbeds. The FI-User selects resources in the resource list, and it returns the confirmation of the reservation as shown in Figure 6. The resources are initiated and provided to fit the F-Interop session.

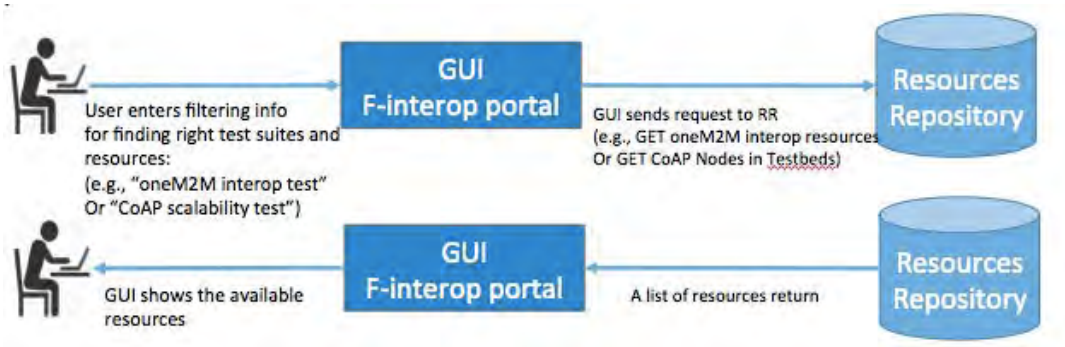

Figure 5 - FI-User Session: Step 1 and Step 2

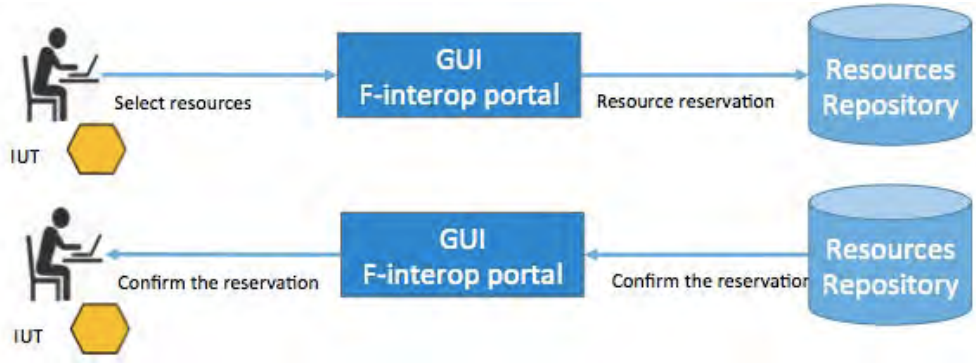

Figure 6 - FI-User Session: Step 3

During the session, the test agent periodically sends RR heartbeats to notify the availability of the resources being used. RR maintains the resource information as "being used" until it receives updated status of the resources or when it doesn't receive the heartbeat for a certain period of times. When the FI-User ends the session, RR receives updated status (release resources) and changes the resource status.

2.3 Implementation decision of F-Interop Resource Repository

As the F-Interop RR aims to connect existing repositories from the three federated testbeds: Fed4Fire, OneLab, IoT-lab, we studied existing data models to find the best solution for designing F-Interop repository. The three testbeds are designed on RSpec, but there is an issue to prevent simple linkage of the repositories from the existing infrastructures. The design of the three testbeds has been started in different objectives and services, and accordingly the depth of the data in repositories is not in the same level. One repository has focused on server level of information (CPU power, processing power, etc.). Two others handle more on experiment node sides but do not have same level of information and do not hold enough details on the type of board, etc. which is important for F-Interop. This barrier has been also detected in Fed4Fire and it has worked on ontology data model for solving the problem. While ontology is a good solution to solve this problem, ontology for IoT has not been tackled in Fed4Fire when F-Interop Resource Repository was designed, or no comprehensive IoT ontology satisfying F-Interop requirements has been developed at the moment. Designing and implementing IoT ontology is not a short-term work with limited resources what current Task 4.1 has, and Fed4Fire+ that just started in January 2017 plans to work on IoT ontology.

Thus, F-Interop decided to implement the resource repository with a rational database considering the project lifetime and service launching plans. Keeping in mind about the linkage problem of the federated testbeds, it particularly focuses on easily extendable data format for the diverse interoperability and performance testing tools that requires different detail hardware and software

information. The data format has been intensively discussed with the consortium partners with several iterations in order to collect comprehensive view to satisfy diverse testing tools. For the linkage with the testbed resources, TBaaS switches RSpec based resource format into JSON that F-Interop is using. All RR operations have a primary key, *resource_id*, which identifies all types of resources (e.g., software resource, hardware resource including both physical and virtual machines).

3 Implementation of F-Interop Resource Repository

The implementation for the repository has been done with MySQL, and the data formats are written in JSON. The messages are designed as simple as possible for easy usage. It uses two messages: *resource_repository.insert_resource* and *resource_repository.get_resource*. Each message has Request and Response messages for querying and answering. The following sections describe the format of APIs and examples of their usages.

3.1 API types and format

All APIs and the messages from/to RR follow the general designing of F-Interop system. RR communicates with other components through F-Interop AMQP Eventbus following its message header format. The body of the message is JSON format and contains details of the resource information. The F-Interop RR basically uses the following two commands for its operations shown in Table 2:

Table 2 - RR Operations and Event

Operations and Event	Description
resource_repository.insert_resource	Message for inserting/updating resources in the RR.
resource_repository.get_resource	Message for querying the RR about resources.
control.resource_repository.error	Message used to indicate errors on coordination component.

The *resource_repository.insert_resource* is used for both inserting and updating resources in RR (including status changes), and *resource_repository.get_resource* is used for querying RR about resources (e.g., resource discovery and resource reservation).

For all Request messages, the following parameters and values are used:

Table 3 - Parameters and Values of Request

Parameters	Value	Description
exchange	'default'	The AMQP exchange
routing_key	'control.resource_repository.service'	The routing key for the RR
reply_to	'control.resource_repository.service.reply'	Direct reply-to (RPC server and client)
correlation_id	str(uuid.uuid4())	The correlation ID

For all Reply messages, the following parameters and values are used:

Table 4 - Parameters and Values of Reply

Parameters	Value	Description
exchange	'default'	The AMQP exchange
routing_key	'control.resource_repository.service.reply'	The routing key for the RR
correlation_id	same as request	The correlation ID

3.1.1 resource_repository.insert_resource

resource_repository.insert_resource is used for registering a new resource or updating an existing resource. The following Table 5 and Table 6 show the format of Request and Response messages of the *resource_repository.insert_resource*, respectively.

Table 5 - Request message of resource_repository.insert_resource

Contents	Type	Value
“_type”	String	“resource_repository.insert_resource”
“resource”	Object	The resource that needs to be inserted/updated. All fields are optional, only the resource_id and the owner_id are mandatory.

Table 6 - Response message of resource_repository.insert_resource

Contents	Type	Value
“_type”	String	“resource_repository.insert_resource”
“ok”	Boolean	true if the resource was inserted/updated correctly, false otherwise.
“error_message”	String	A message describing the error (present if ok = false)
“error_code”	Integer	Present if ok = false

The following is an example of a Request message of inserting a resource:

```
---
ROUTING_KEY: control.resource_repository.service
---
HEADERS: None
---
PROPS: {
   "correlation_id": "1a58de57-176e-41c4-bca3-e81b968142c9",
   "content_type": "application/json",
   "reply_to": "control.resource_repository.service.reply",
   ...
}
---
BODY {
   "_type": "resource_repository.insert_resource",
   "resource":{
      "resource_id": "unique-resource-identifier",
      "privacy_flag": "public",
      "available": true,
      "owner_id": "unique-owner-identifier",
      "ts_creation": 1256953732,
      "ts_update": 1256953732,
      "hardware": {
         "manufacturer": "HW manufacturer",
         "machine_type": "phy",
         "hw_platform": "HW platform (board name)",
         "power_supply": "Power supply",
         "processor": "Processor name",
         "cpu_cache": "CPU cache",
         "ram": "RAM",
         "disk": "Secondary storage info",
         "supported_medias": [
            {
               "name": "Media name",
               "version": "Media version",
               "network_card": "Network card info",
               "mac_address": "aa:bb:cc:dd:ee:ff",
               "if_bandwidth": "Interface bandwidth"
            } ],

         "other_hardware_info": [
```

```
            {"parameter": "info_key", "value": "info_value"}
           ] },

       "fw_os": {
          "name": "OS/Firmware name",
          "version": "OS/Firmware version",
          "ip_address": "IP address",
          "supported_protocols_services": [
             {
                "name": "Protocol name 1",
                "version": "Protocol version 1",
                "port": 123,
                "other_protocol_info": [
                   {"parameter": "info_key", "value": "info_value"}
                ]
            } ],

"other_fw_os_info": [
             {"parameter": "fw_os_info", "value": "fw_os_value"}
          ]
       },
       "software": {
          "name": "SW name",
          "version": "SW version",
          "role": "client",
          "automatic_flag": "auto",
          "test_environment": "central_server",
          "testing_protocol": {
             "name": "Protocol name 2",
             "version": "Protocol version 2",
             "port": 1234,
             "other_protocol_info": [
                {"parameter": "info_key", "value": "info_value"}
             ]
          },
          "testing_tool": [
             {
                "name": "Testing Tool name",
                "version": "Testing Tool version"
                                } ],

          "other_software_info": [
             {"parameter": "info_key", "value": "info_value"}
          ] },

       "location": {
          "latitude": 11,
          "longitude": 22,
          "country": "CH",
          "city": "City",
          "address": "Address",
          "x": 0,
          "y": 0,
           "z": 0 }
           }
      }
}
---
```


The following is an example of a Reply message of inserting a resource:

```
-- -
ROUTING_KEY: control.testcoordination.service.reply
-- -
HEADERS: None
-- -
PROPS: {
   "correlation_id": "1a58de57-176e-41c4-bca3-e81b968142c9",
   "content_type": "application/json",
   "reply_to": null,
...
}
-- -
BODY {
   "_type": "testcoordination.testsuite.getstatus",
   "ok": true
}
-- -
```

3.1.2 resource_repository.get_resource

resource_repository.get_resource is used for discovering or querying resources. The following Table 5 and Table 6 show the format of Request and Response messages of the resource_repository.get_resource, respectively.

Table 7 - Request message of resource_repository.get_resource

Contents	Type	Value
“_type”	String	“resource_repository.get_resource”
“query”	Object	An object matching the structure of a resource object. All fields that are present in the query will be used to match resources. All resources that match all the fields present in the query will be returned (see example).

Table 8 - Response message of resource_repository.get_resource

Contents	Type	Value
“_type”	String	“resource_repository.get_resource”
“ok”	Boolean	True if the query was executed correctly, false otherwise.
“results”	Array of Objects	The list of resources that match the query. An empty list if no result. (Present if ok = true)
“error_message”	String	A message describing the error (present if ok = false).
“error_code”	Integer	Present if ok = false.

The following example shows discovery a resource by sending RR resource_repository.get_resource Request message. This query returns all public resources that match all the specified fields. In other words: all public resources that support 802.11n WiFi, that have CoAP observe testing tool and that are in Switzerland, for instance.

```
-- -
ROUTING_KEY: control.resource_repository.service
-- -
HEADERS: None
-- -
PROPS: {
   "correlation_id": "1a58de57-176e-41c4-bca3-e81b968142c9",
   "content_type": "application/json",
```

```
   "reply_to": "control.resource_repository.service.reply",
   ...
}
-- -
BODY {
   "_type": "resource_repository.get_resource",
   "query": {
      "hardware": {
         "supported_medias": [
{
   "name": "2.4GHz WiFi",
   "version": "802.11n"
}
] },
      "software": {
         "testing_tool": [
            {
               "name": "CoAP observe",
               "version": "RFC 7641"
} ]
      },
      "location": {
         "country": "CH"
      }
} }
-- -
```

The following message is the RR's response on the Request message shown above.

```
-- -   ROUTING_KEY: control.testcoordination.service.reply
-- - HEADERS: None
-- - PROPS: {
   "correlation_id": "1a58de57-176e-41c4-bca3-e81b968142c9",
   "content_type": "application/json",
   "reply_to": null,
   ...
}   -- -
BODY {
   "_type": "resource_repository.get_resource",
   "ok": true,
   "results": [
      {
         "resource_id": "resource-001-unique-id",
         "owner_id": "user-001-unique-id",
         "privacy_flag": "public",
         "available": true,
         "ts_creation": 1481382022000,
         "ts_update": 1481382022000,
         "hardware": {
            "hw_platform": "HW platform info",
            "manufacturer": "Manufacturer info",
            "machine_type": "phy OR vm",
            "power_supply": "Power supply info",
            "processor": "Processor info",
            "cpu_cache": "CPU cache info",
            "ram": "RAM info",
            "disk": "Secondary storage info",
           "other_hw_info": [
           {"param": "Extra info name", "value": "Extra info value"},
           ... ],
   "supported_medias": [
      {
```

```
            "name": "2.4GHz WiFi",
            "version": "802.11n",
            "network_card": "None",
            "mac_address": "aa:bb:cc:dd:ee:ff",
            "if_bandwidth": "150 Mbit/s"
        },
       ...
] },
    "fw_os": {
    "name": "Firmware/Software name",
    "version": "1.2.3",
    "ip_address": "192.168.1.100",
    "supported_protocols_services": [
        {
            "name": "CoAP observe",
            "version": "RFC 7641",
            "port": 5683,
            "other_protocol_info": [
                {"param": "Extra info name", "value": "Extra info value”},
            ... ]
    }, ]
    "other_fw_os_info": [
        {"parameter": "fw_os_info", "value": "fw_os_value"},
        ...
      ] },
"software": {
    "name": "SW name",
    "version": "SW version",
    "role": "client",
    "automatic_flag": "auto",
    "test_environment": "central_server",
    "testing_tool": [
        {
            "name": "Tool name",
            "version": "1.2.3"
       } ]
    "testing_protocol": {
        "name": "CoAP observe",
        "version": "RFC 7641",
        "port": 5683,
        "other_protocol_info": [
            {"param": "Extra info name", "value": "Extra info value"},
            ...
          ] },
                "other_software_info": [
                    {"param": "Extra info name", "value": "Extra info value"},
      ... ] },
            "location": {
                "country": "CH",
                "city": "Geneva",
                "address": "None",
                "latitude": 0,
                "longitude": 0,
                "x": 0,
                "y": 0,
                "z": 0
}
      },
     {
           "resource_id": "resource-123-unique-id",
```

```
            "owner_id": "user-002-unique-id",
            "privacy_flag": "public",
            "available": true,
            "ts_creation": 1481382022000,
            "ts_update": 1481382022000,
            "hardware": {
            ... },
           "fw_os": {
               ...
           },
           "software": {
            ... },
           "location": {
               ...
            }
       },
... ] }
```


It is noted that the FI-USER interaction for the RR operations will be demonstrated in T4.3. This document is to provide the RR operations, APIs, and the message formats of RR. It is also noted that the other APIs for F-Interop platform are provided in http://doc.f-interop.eu. The document states all F-Interop APIs per components with message types and formats.

4 Conclusion

For building F-Interop Resource Repository aiming to connect existing repositories from the three federated testbeds: Fed4Fire, OneLab, IoT-lab, intensive studies have been made on the existing data models to find the best solution for designing F-Interop repository (that short summary is described in Appendix). After checking various data models and well-known APIs, the consortium had put severe efforts to build the most suitable data format that can satisfy diverse types of different testing tools and are easily extendible for the further cases.

The Resource Repository has been designed aligned with F-Interop Architecture described in D1.3, and carefully followed the requirements and session steps that described in D1.1. It collected all requirements from WP2 and WP3 for the specification of the resource information. The implementation and its integrated tests have been performed with T4.2 and T4.3. The implemented RR has been tested in several iterations integrated with other core components such as TBaaS and F-Interop GUI, and verified its functions and behaviours. Although the Task duration is until M18, it will be continuously maintained and updated during the remaining project lifetime as needed.

Appendix. Survey on data models and APIs

The Appendix provides the list of data models that we studied in the designing phase of F-Interop resource repository.

A.1 Fed4Fire APIs

A.1.1 RSpec

RSpec, or "Resource Specification" documents are XML documents following agreed schemas, used to describe resources. There are three different types of RSpec, each used to describe resources when communicating with an AM. The communication with an AM, is based on the common AM API that requires AMs to communicate using RSpec data types.

- Advertisement RSpec: This is the document that is returned by an AM that describes the resources that the AM has.
- Request RSpec: This is the document that a user sends to an AM to describe the resources that he wants to reserve.
- Manifest RSpec: This is the document returned by an AM that describes the resources that a user has reserved at an AM.

Key features of RSpec are as followings:

Identifying resources:

- All nodes and links are identified by URN

Nodes

- Node have types
- Geographic information can be attached
- Virtualization technology is included as a field
- Descriptions of additional optional facilities provided by the component manager (e.g. login protocols, installation of software, commands to run) can be supplied

Links

- Links are point-to-point: LANs and other "full connectivity" environments (such at the Internet), a "LAN node" is created, and all members are linked to it.
- Links have bandwidth, a type, etc.
- Links endpoints reference Interfaces on Nodes
- Links have LinkTypes which describe how experimental traffic is to be encapsulated and what layer(s) are supported for that traffic.

Interfaces

- Endpoint of a link
- Named by node, plus an opaque interface name
- In progress: Interfaces will be first-class entities, declared as part of the component they belong to

Metadata

- A "valid until" field
- A "generated" time

ExternalReferences

- A mechanism by which CMs can describe how their components connect with the components in other CMs.

Wireless Nodes

- We need to make special provision for nodes which can communicate wirelessly: WirelessRSpec

Extensions (see RSpecExtensions)

- The RSpec must provide anextensions mechanism to accommodate the wide range of stakeholders which will make use of the RSpec.

A.1.2 OMF

OMF[1] is a control, measurement and management framework for experimental platforms (aka testbeds) which allows the definition and orchestration of experiments using shared (already provisioned) resources from different federated testbeds. OMF was originally developed for single testbed deployments, but has recently been extended to support multiple deployments and the following features. First, it provides a domain-specific language based on an event-based execution model to fully describe even complex experiments (OEDL). OMF also defines a generic resource model and concise interaction protocol (FRCP), which allows third parties to contribute new resources as well as develop new tools and mechanisms to control an experiment (as mentioned on the previous slide). It has a distributed communication infrastructure based on XMPP supporting the scalable orchestration of thousands of distributed and potentially disconnected resources.

A.1.3 RDF

Resource Description Framework *(*RDF) is a framework for representing information in the Web. RDF has features that facilitate data merging even if the underlying schemas differ. RDF extends the linking structure of the Web to use URIs to name the relationship between things as well as the two ends of the link (this is usually referred to as a "triple"). Using this simple model, it allows structured and semi-structured data to be mixed, exposed, and shared across different applications.

RDF Schema (RDFS) provides a data-modelling vocabulary for RDF data. RDF Schema is a semantic extension of RDF. It provides mechanisms for describing groups of related resources and the relationships between these resources.

A.2 Survey on Data/Resource Models for IoT

A.2.1 Open Geospatial Consortium (OGC) SensorML

OGC developed Sensor Model Language (SensorML) that provides descriptions of sensors and sensor systems for inventory management. It also provides sensor and process information in support of asset and observation discovery, performance and quality of measurement characteristics (e.g., accuracy, threshold, etc.),22 general descriptions of components (e.g. a particular model or type of a sensor) as well as the specific configuration of that component when its deployed, 22 a machine interpretable description of the interfaces and data streams flowing in and out of a component, 22 an explicit description of the process by which an observation was obtained (i.e., it's lineage) and 22an executable aggregate process for deriving new data products on demand (i.e., derivable products). It supports the processing and analysis of the sensor observations and the geolocation of observed values (measured data), and archeive the processing and analysis of the sensor observations 22 Support the geolocation of observed values (measured data)

OCG also defines the Sensor Web Enablement (SWE) Common Data Model Encoding Standard. It defines low level data models for exchanging sensor related data between nodes of the OGC Sensor Web Enablement (SWE) framework. SWE Common Data Models aims to achieve interoperability both in the syntactic level and at the semantic level by using ontologies and semantic mediation.

The inter-dependency of the SensorML and SWE Common Data Model is as following:

[1] http://omf.mytestbed.net

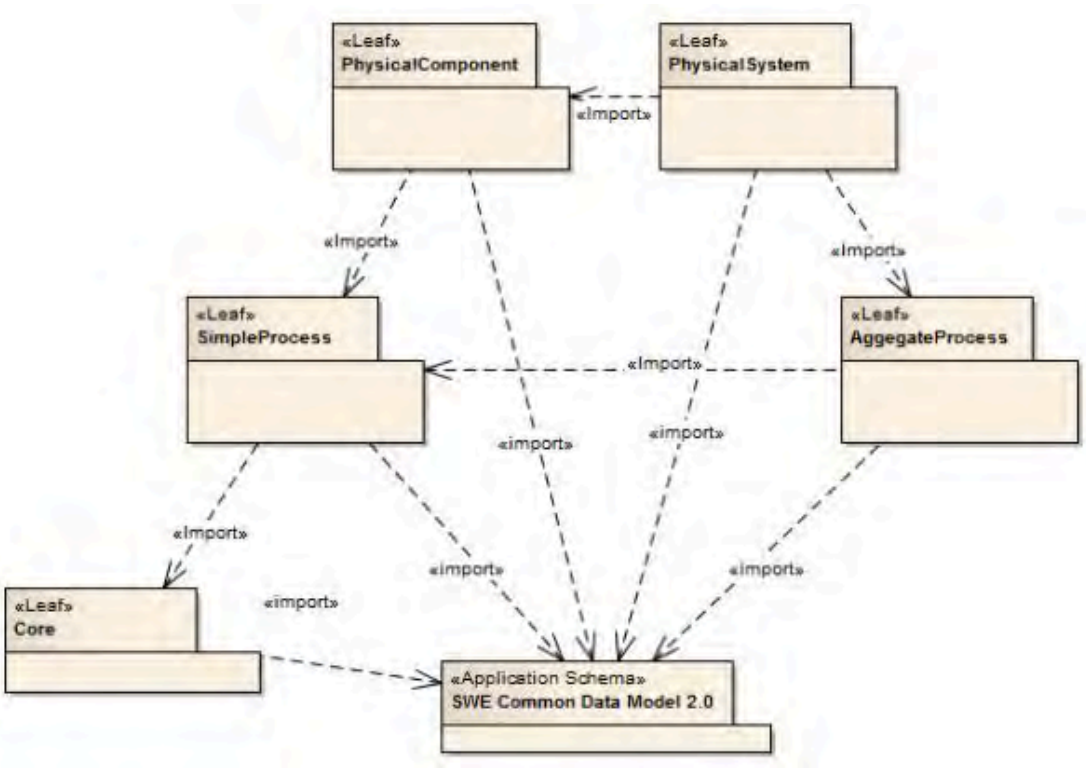

Figure 7 - inter-dependency of SensorML and SWE common data model (*source: OGC*)

A.2.2 OMA LWM2M

Open Mobile Alliance (OMA) develops Lightweight Machine to Machine (LWM2M). LWM2M is used for device management operations, and its Object Model is being used to provide a resource model foo Applications. IPSO Alliance is defining IPSO Objects built on the LWM2M Object Model for smart city/building applications.

The LWM2M Enabler defines a simple resource model where each piece of information made available by the LWM2M Client is a Resource. The Resources are further logically organized into Objects. The LWM2M Client can have any number of Resources, each of which belongs to an Object.

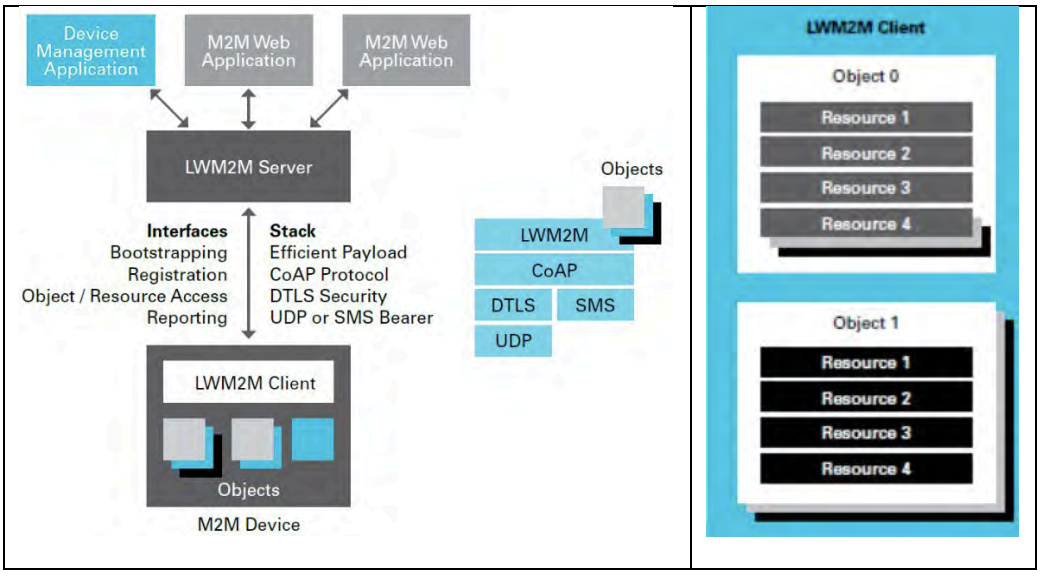

Figure 8 - LWM2M building blocks (*source: LWM2M*)

OMA Object/Resource Template [2]

Object definition:

Name	Object ID	Instances	Mandatory	Object URN
Object Name	16-bit Unsigned Integer	Multiple/Single	Mandatory/Optional	urn:oma:lwm2m:{oma,ext,x}:{Object ID}

Resource definition:

ID	Name	Operations	Instances	Mandatory	Type	Range or Enumeration	Units	Description
0	Resource Name	R (Read), W (Write), E (Execute)	Multiple/ Single	Mandatory/Optional	String, Integer, Float, Boolean , Opaque, Time, Objlnk none	If any	If any	Description

Executable Resource Arguments Definition

ID	Resource Name	Order	Name	Type	Range or Enum	Unit	Description
id	Resource name	[0:9]	String	LWM2M Data Types	If any	If any	Description

Execution Resource Arguments definitions

ID	Resource Name	Order	Name	Type	Range or Enum	Unit	Description
5	Delete	0	-	none	-	-	1 argument EXECUTE /X/0/5 0
7	Update	0	Remove	none	-		2 arguments Ex EXECUTE /X/0/7 0,1='2'
		1	Keep	Integer	[0-2]		
10	Create						

A.2.3 Open Connectivity Foundation (OCF)[3] Resource Model

OCF develops a RESTful architecture that would limit the system to just five (CRUDN[4]) APIs and a constructive data model. OIC Core specification and Resource type specification provide information on the format of resource type, resource management and resource discovery.

---

[2] "OMA Lightweight Machine to Machine Protocol v1, 0"
http://technical.openmobilealliance.org/Technical/release_program/lightweightM2M_v1_0.aspx

[3] Formerly Open Interconnect Consortium (OIC)

[4] Simple Request/Response mechanism with Create, Retrieve, Update, Delete and Notify command

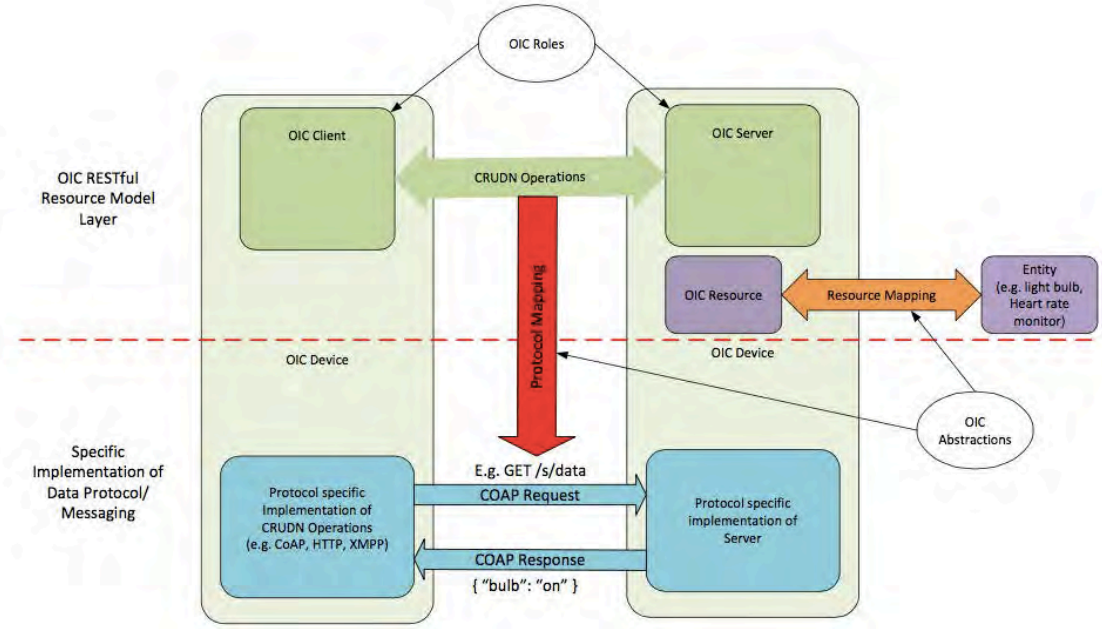

Figure 9 - OIC architecture - concept (*source:OIC core specification*)

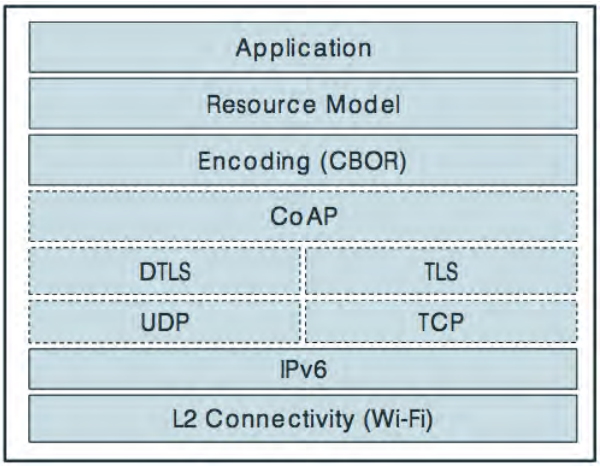

Figure 10 - OIC Stack

Figure 11 - Device specification: smart home device use case

All Resources in OCF conform to Resource Types for vertical application/business domains like Smart Home, Industrial, Healthcare, Automotive and others.

Device Type	Minimum Resource Set
Air Conditioner	Binary Switch, Temperature
Air Purifier	Binary Switch
Blind	Open Level
Dishwasher	Binary Switch, Mode
Door	Open Level
Clothes Dryer	Binary Switch, Mode
Clothes Washer	Binary Switch, Mode
Fan	Binary Switch
Garage Door	Door
Light	Binary Switch
Oven	Binary Switch, Temperature (2)
Printer	Binary Switch, Operational State

Device Type	Minimum Resource Set
Refrigerator	Binary Switch, Refrigeration, Temperature (2)
Robot Cleaner	Binary Switch, Mode
Smart Plug	Binary Switch
Switch	Binary Switch
Thermostat	Temperature (2)
Camera	Media
Generic Sensor	Sensor
Receiver	Binary Switch, Audio, Media Source List (2)
Scanner	Binary Switch, Operational State, Automatic Document Feeder
Security Panel	Mode
Television	Binary Switch, Audio, Media Source List
Water Valve	Open Level

Resource Types	Use Case
Air Flow	Indoor Environment Control
Air Flow Control	
Battery	Device Control
Binary switch	Device Control
Brightness	Lighting Control
Colour Chroma	
Colour RGB	
Dimming	
Door	Indoor Environment Control
Energy Consumption	Energy Management
Energy Usage	
Humidity	Indoor Environment Control
Icemaker	Device Control

Resource Types	Use Case
Lock	Keyless Entry
Lock Code	
Mode	Device Control
Open Level	
Operational State	
Ramp Time	Lighting Control
Refrigeration	Device Control
Temperature	Indoor Environment Control
Time Period	Device Control

Resource Type	Use Case
Audio	TV, Home Entertainment
Auto Focus	IP Camera
Auto White Balance	IP Camera
Automatic Document Feeder	Scanner Support
Button	Device Control
Colour Saturation	IP Camera
DRLC	Smart Energy
Energy Overload	Smart Energy
Media	IP Camera
Media Source List	TV, Home Entertainment
Movement (Linear)	Robot Cleaner
Night Mode	IP Camera
PTZ	IP Camera
Signal Strength	Proximity

Sensor Support Resources

Sensor Resource Type	Use Case
Acceleration	Extended Sensor Set (for a Generic Sensor Device)
Activity Count	
Atmospheric Pressure	
Carbon Dioxide	
Carbon Monoxide	
Contact	
Glass Break	
Heart Rate Zone	
Illuminance	
Magnetic Field Direction	
Presence	
Radiation (UV)	
Sleep	
Smoke	
Three Axis	
Touch	
Water	

A.3 IoT Ontology

A.3.1 W3C Semantic Sensor Network Ontology5 (OWL 2)

The W3C SSN-XG (Semantic Sensor Networks Incubator Group)[6] worked from March 2009 to September 2010 was initiated by the CSIRO and Wright State University as a forum for the development of an OWL ontology for sensors and to further investigate annotation of and links to existing standards.

In this group, Semantic Sensor Network Ontology (SSNO) has been developed to help process and understand sensor information, and to allow the discovery, understanding, and querying of sensor data. The following figure shows the key concepts and the relations among the conceptual modules.

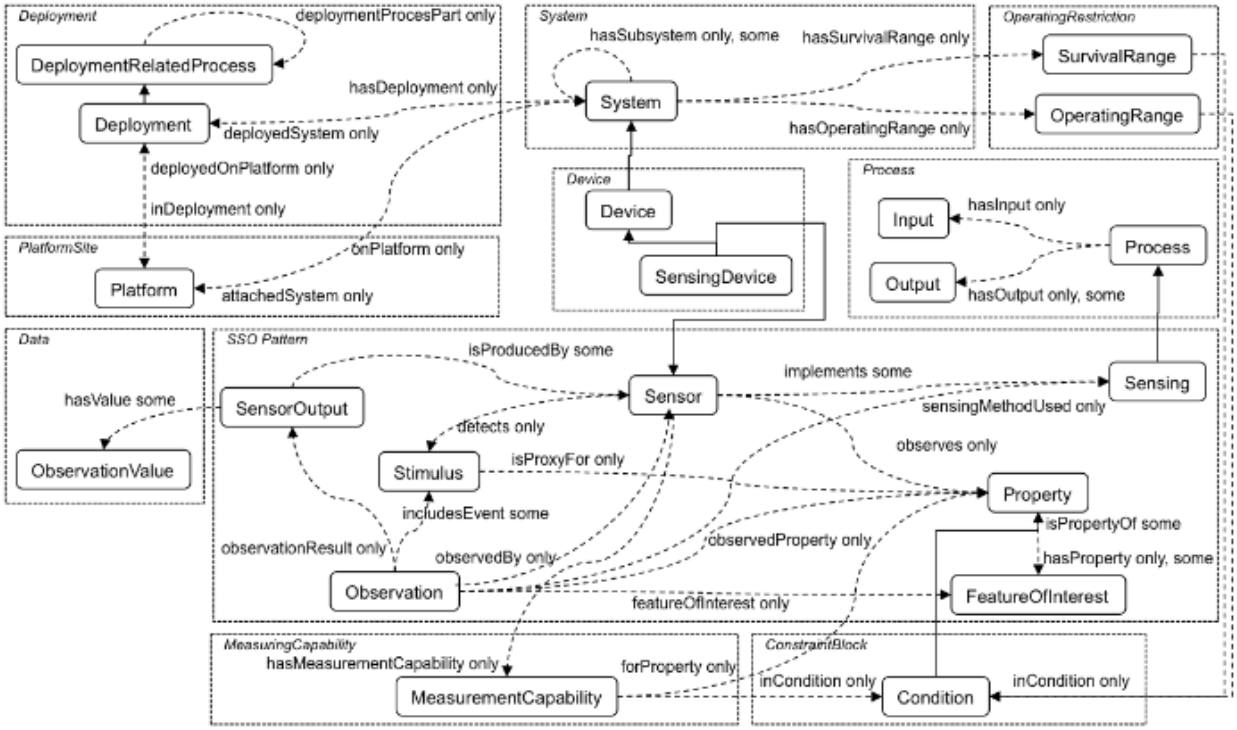

Figure 12 - SSN ontology of SSNG (*source:SSNG*)

The following figure shows Ontology view showing systems, deployments, platforms and operating and survival conditions.

5 W3C Semantic Sensor Ontology: http://www.w3.org/2005/Incubator/ssn/XGR-ssn-20110628/

6 http://www.w3.org/2005/Incubator/ssn/wiki/Main_Page

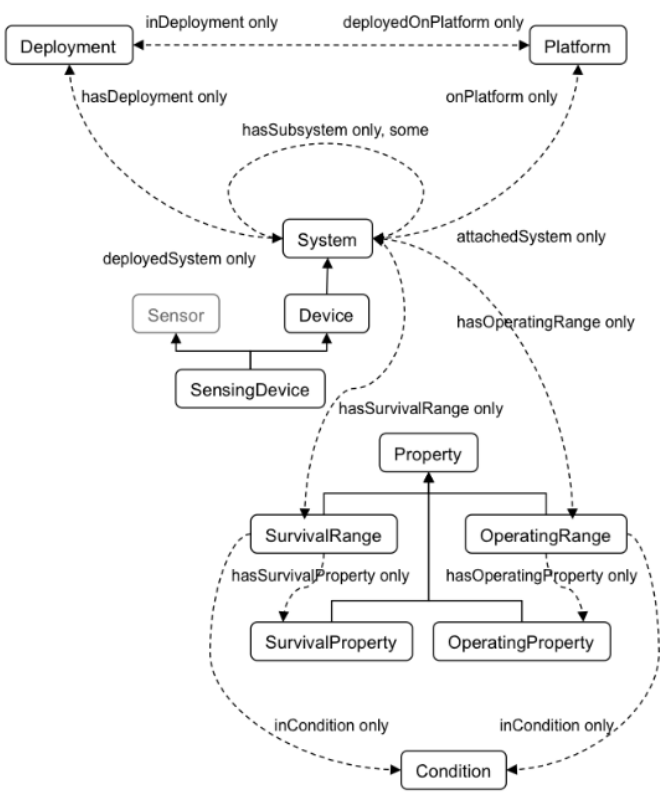

Figure 13 - SSN ontology of the Semantic Sensor Networks Incubator Group

The main use cases of the SSN ontology are data discovery and linking, device discovery and selection, provenance, and device operation, tasking and programming — cover a range of applications from data and linked open data (LOD) to selection and deployment and mission planning.

A.3.2 OneM2M Base Ontology

TS-0012-V-0.10-0 states “oneM2M's base ontology” that is the only ontology document by OneM2M. It is to provide syntactic and semantic interoperability of the oneM2M System with external systems.

Table 9 - Example for classes and properties mapping between two ontologies (*source: TS-0012-V0.10-0*)

Properties mapping			Classes mapping		
property I	mapping relationship	property II	class I	mapping relationship	class II
OntologyB: hasSwitch	rdfs:subPropertyOf	OntologyA: hasOperation	OntologyB: appliance	rdfs:subClassOf	OntologyA:device
OntologyA: hasPower	owl:equivalentProperty	OntologyB: hasPower	OntologyB:lamp	owl:equivalentClass	OntologyA:light
OntologyA: hasVendor	owl:equivalentProperty	OntologyB: hasManufacturer	OntologyB:Switch	rdfs:subClassOf	OntologyA:Operation

Table 10 - Example for individual annotation between two ontologies (*source: TS-0012-V0-10-0*)

Individuals	Semantic annotation based on Ontology A		Semantic annotation based on Ontology B	
	Properties	classes	properties	Classes
Light A	rdf:type	Ontology A: Light	rdf:type	Ontology B:ledLight
	OntologyA: hasOperation	Ontology A:Open	OntologyB: hasColor	rdf:datatype="&xsd;string">'red'<
	OntologyA: hasStatus	rdf:datatype="&xsd;boolean">true<	OntologyB: hasSwitch	OntologyB:Switch
NOTE: The two methods can be used jointly or independentlly.				

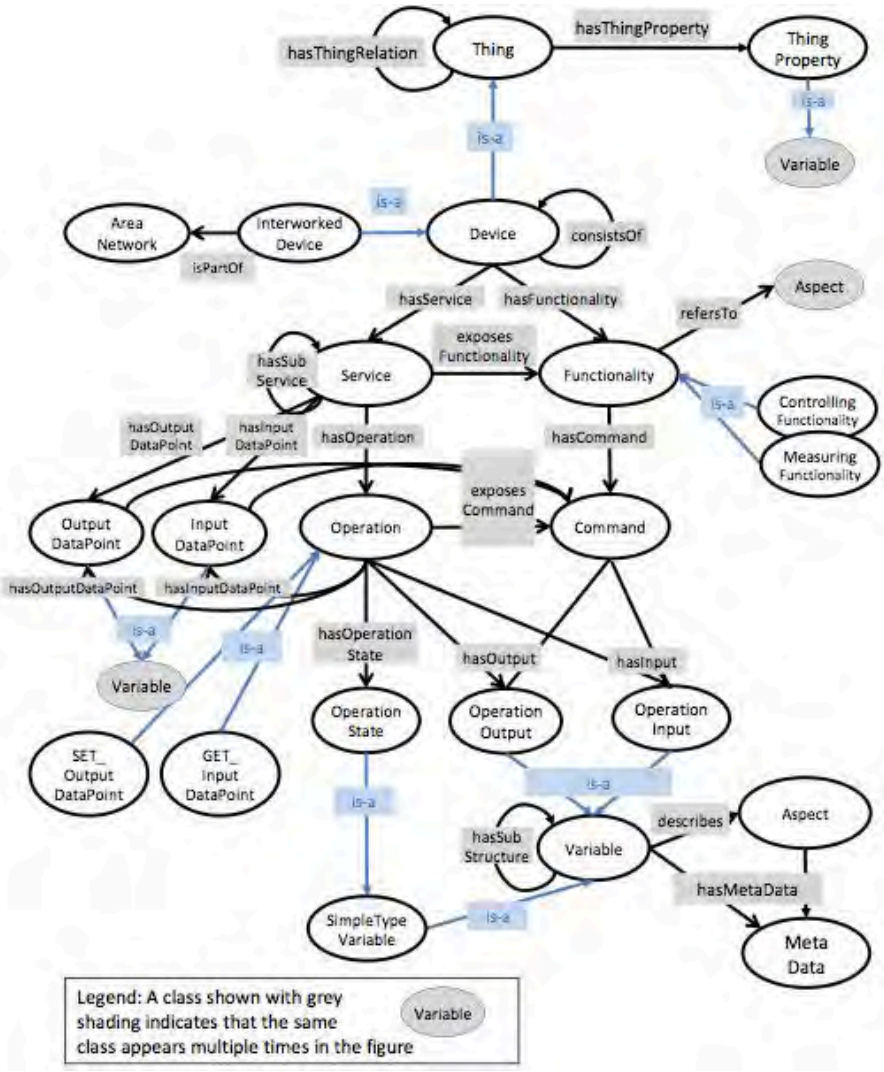

Figure 14 OneM2M base ontology (*source: TS-0012-V-0.10-0*)

A.3.3 Gap of the current study of IoT ontology

SSNO (Semantic Sensor Network Ontology) is a base of the most existing studies on IoT resource modelling that focus on sensors and sensor networks. Current IoT related ontologies are not yet fully connecting diverse types of IoT devices (e.g., Servers, IoT gateway, sensor nodes, actuator, etc). Thus, it is necessary to design of IoT resource modelling that provide full information of IoT devices with its relationship with other IoT devices.