Automating Heterogeneous Internet of Things Device Networks from Multiple Brokers with Multiple Data Models

Pierfrancesco Bellini
Distributed Systems and Internet Technology
Lab DISIT, Florence, Italy
0000-0002-8167-1003

Chiara Camerota
Distributed Systems and Internet Technology
Lab DISIT, Florence, Italy
0000-0002-0363-2404

Paolo Nesi
Distributed Systems and Internet Technology
Lab DISIT, Florence, Italy
Paolo.nesi@unifi.it ,
https://www.snap4city.org
0000-0003-1044-3107

Abstract — **The Internet of Things (IoT) is becoming pervasive and with each new installation of the IoT platform legacy internal and external brokers have to be integrated. Internal brokers are those under the control of the platform, while external brokers are managed by third parties. Both brokers kind may be multiservice / multi-tenant and may manage multiple Data Models. The interoperable management of these complex networks has to pass from the IoT device registration which is typically a re-current operation since the IoT networks are in continuous evolution. In this paper, the above-mentioned problems have been addressed by the introduction of our concept of IoT Directory and reasoning tools to (i) manage Internal and External brokers, (ii) perform the automated registration by harvesting and reasoning of devices managed into external brokers single- or multi-tenant services, (iii) perform the automated registration and management of Data Models, and any custom Data Model. The solution has been developed and tested into Snap4City, an 100% open-source IoT platform for Smart Cities and Industry 4.0, the official FIWARE platform, EOSC, and lib of Node-RED. The specific IoT Directory has been developed in the context of the Herit-Data Project, the results have been validated in wide conditions of the whole Snap4City network of more than 18 tenants, and billions of data.**

Keywords— Automated IoT (Internet of Things) Device Registration, Internal and External IoT (Internet of Things) Brokers, IoT (Internet of Things) Network, Smart Data model, Snap4city

I. INTRODUCTION

The Internet of Things (IoT) defines a paradigm for the computation and communication among things that everyone uses more and more daily [1]. It is due to the intense deployment campaign worldwide about Low-Power Wide Area Network technologies [2]. Nowadays, the real world and IoT devices are integrated tother with some humans in the loop. Communication among devices may support various protocols (e.g., Message Queue Telemetry Transport or MQTT, Next Generation Service Interfaces or NGSI, Advanced Message Queuing Protocol or AMQP, Constrained Application Protocol or CoaP) thus, the cloud-fog infrastructures are exploited, as well as the management of the information [3]. In terms of data management, the complexity is growing, not only for the huge amount of data but also for the need of interoperability and abstraction. The **Gateway** concept is a relevant entity to manage IoT devices into an **IoT Platform**. It may be integrated with one or several **IoT Brokers** to send/receive data to/from devices. The Gateways and its IoT Brokers are typically based on a single protocol and managed by third parties as a public service for several customers interested in the same area (for example LoraWAN services [4]). A Gateway may abstract from the IoT Broker level managing them for multiple organizations/tenants (which can be regarded as customers of Gateway services to manage a number of **IoT Devices**), via some API and/or Web user interface. Typical IoT Brokers are capable to manage only one organization, and thus they are single-tenant, in the sense that they broker messages using the topic concepts (which can be regarded as the key for subscription) without any internal partition of services as a sort of family of devices and subscriptions. Some IoT Brokers can be multi-tenant, such as the FIWARE **Orion Broker**, which provides a partition of the devices in groups, and each of them may have a dedicated service/path for a specific scope (or of a specific customer). Furthermore, devices of different tenants could exist physically in different places (even having identical IDs), and the subscription to the broker's tenant may imply getting all messages/services in the partition. That is feasible only if the subscriber knows the identifier of the service path and, in the cases of access control, has the grant to access at the services. The complexity of **IoT Platforms** grows with the need of managing multiple IoT Brokers which can be managed by third parties different from the IoT Platform manager, i.e., **IoT External Brokers**, and/or directly managed by the IoT Platform-tools, i.e., **IoT Internal Brokers**, can be adopting different protocols, formats.

For the **IoT External Brokers**, the entities (IoT Devices) are directly registered on the IoT Broker which is not under the control of the IoT Platform. Thus, the IoT Platform does not know the IoT Device data structure nor the composition of messages and services. Most of the IoT Platforms neglect these interoperability and integration aspects and provide simplified solutions. They do not care about Internal/External Brokers, just providing the possibility to set up end-to-end solutions with some restricted usage, for example using only internal brokers they provide. Thus, AWS IoT by Amazon (AWS) [5] and Siemens MindSphere [6] make the broker structure transparent to their users unless they buy a specific add-on. While IoT Platform like Google IoT Cloud (Google IOT) [7] shows the Broker architecture but allows the usage of only one kind of protocol (e.g., MQTT). At least, solutions like MS Azure IoT (MS Azure) [8] or IBM Watson [9] are more flexible. MS Azure does not provide to cluster their objects, in other words, supporting only one organization on broker; the IBM solution does not allow the connection of External Brokers. In summary, most of the solutions provide

simple scenario, and mainly assume to have customers starting to use their solution from scratch (on cloud or on premise), offering limited capabilities to deeply integrate the platform with legacy IoT Broker and Network. Nevertheless, all of them provide the possibility of connecting to other IoT Brokers and Network by means of REST Call on API. Naturally, in those scenarios, the third-party brokers are not directly connected and neither managed in terms of IoT Device registration, subscription, data storage, search, etc.

Different IoT Devices connected to the same broker adopt the same protocol and may use different data models. If the message format is based on JSON, the corresponding schema may be defined and used for validation, while variables/attributes can be differently defined. For example, FIWARE Orion Broker adopts the NGSI protocol with the possibility of managing the so-called **Smart Data Models** from which IoT Devices can be templated out [10]. The IoT Platform or Gateway must recognize these **IoT Device Models** and manage them (registering, processing, producing, storage, etc.). In the case of **IoT External Brokers**, the IoT Platform may not know the IoT device models, and neither the identifier (topic) of the External IoT Devices. Hence, at the arrival of a message from an unknown device (which can partially provide information in its body, typically not the metadata, since most of the devices minimize the data transmission), the IoT Platform is not in the condition of registering the device, and neither to correct link the message to the former devices. Thus, the devices and messages are easily managed when a new external device is added to the IoT Platform if the data model adopted is known. In other words, if the IoT Platform knows the data model adopted by a device it is easier to identify the data structure and so it could automatically manage the relationships among data entities and verify coherence. For all these reasons, the Data Model concepts and formalisms are crucial.

In this paper, the above-mentioned problems and other aspects have been addressed to design and implement a solution for leveraging IoT network interoperability and the management of Data Models. To this end, we have created the concept and tool named **IoT Directory** in the Snap4City architecture [11]. The **IoT Directory** supports: (i) **Internal** and **External** brokers, (ii) the automated registration of devices managed into External Brokers single- or multi-tenant services, (iii) automated registration by harvesting and reasoning of IoT Devices compliant with standard models such as FIWARE Smart Data Model, and any custom Data Model in Snap4City IoT Device Model providing a formal semantic definition of attributes. The research presented in this paper has been developed for Snap4City architecture presently quite diffuse in Europe as 100% open source IoT platform for Smart Cities and Industry 4.0 and it is an official FIWARE platform and solution, and of EOSC, Node-RED [12], [13]. The specific IoT Directory has been developed in the context of Herit-Data Project which promotes the use of smart and open data to better manage tourism flows in natural and cultural heritage sites, the results have been validated in wide condition of the whole Snap4City network of more than 18 tenant, and billions of data.

The paper is organized as follows. Section 2 presents the major requirements for data model and broker interoperability in IoT Platforms. Section 3 shows the role of IoT Directory in IoT architecture for managing internal/external brokers with the aim of automated registration, management vs data models and formal definition of their attributes. In Section 4, some details regarding the validation of the solution are reported. The validation included verifying the processing timing and giving a general numeric KPI about the shape of the entities. In Section 5, the conclusions are drawn.

II. Requirements Analysis and Related Work

In this section, the requirements that an IoT Platform for IoT Network management and exploitation should satisfy are reported and commented on. They have been identified in the context of workshops, conventions, and analysis for the development and exploitation of the Snap4City platform covering smart city and Industry 4.0 domains. The requirements for the IoT Platform are presented in logical order from R1 to R10 as follows. The following list of the requirement refer also to a set of well-known platforms: AWS, Google IOT, MS Azure, Siemens MindSphere and IBM Waston. Therefore, an **IoT Platform** should provide support to:

1. **Manage different kinds of IoT Brokers, IoT Devices and IoT Edge Devices**. They should be based on different protocols, formats, and modalities to establish connections with the IoT Platform. Focusing on the Platform considered, all of them support MQTT and HTTP, while Google IoT and Azure IoT support only MQTT Broker. It is important to highlight that most of the platforms provide specific components for different protocols, for instance: Amazon MQ that supports Broker with AMPQP, MQTT, OpenWire and STOMP protocol.
2. **Connect External and Internal Brokers.** They could be multiservice and could provide different protocols. Internal Brokers should be deployed and registered by the IoT Platform, while the External Brokers would be only registered to use them. In the Platform considered, the brokers are the products' core of stakeholders offers, for this reason the requirement is partially satisfied. In that sense, AWS IoT and Siemems MindSphere offer a paid add-on.
3. **Register, manage and use messages conformant to any Data Model with any data type**. Providing, receiving, managing, storing, and retrieving messages for any IoT Device of any Data Model with its attributes and data types, and related access control. A Data Model provides a model format for IoT Device messages with several variables/parameters/attributes with their specific data types. For the listed Platforms, the messages from the IoT Devices are freely shaped, so to assure the data flexibility to the detriment the data model. For example, Google IoT and IBM Watson use formats as JSON or XML.
4. **Verify the correctness of IoT Messages of IoT Devices.** The platform should be capable of verifying

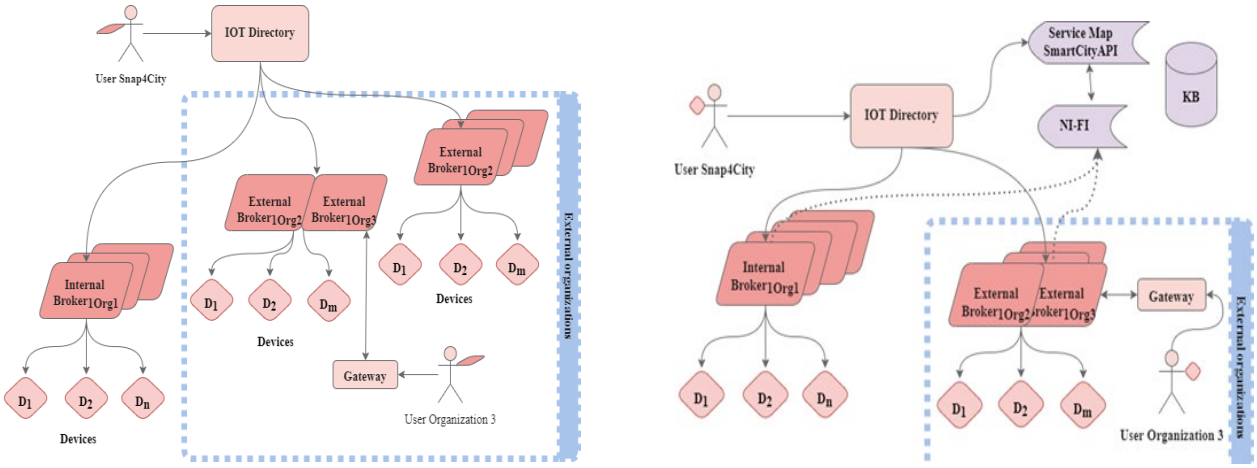

Figure 1. (left) registration of an IoT Broker, (right) registration of an IoT Device. The solid lines indicate the registrations, while the dashed lines indicate the data flow of the subscriptions.

correctness of messages in terms of model and format including verification at level of attributes, before accepting/sending them. Please note: this requirement is satisfied by each solution considered since the IoT Devices are formally defined at the registration phase.

5. **Semantic Interoperability.** This requirement is fundamental to achieve the coherence among different IoT Devices (e.g., provided by different builders, addressing the same concepts, information on attributes). An IoT Platform should be capable to recognize/classify/retrieve information/attributes and behave accordingly to the semantic data model and types. For example, an IoT application should not risk misunderstanding the unit of measure assigned to attributes of different devices which have the same name, but different units.
6. **Support automatics deploy of Internal IoT Brokers.** The IoT Platform should provide support for the automated deployment of IoT Broker internally managed. And thus, Internal Brokers are directly managed by the Platform which directly performs the registration of IoT Devices on them. The result is an easy experience for the user and an easy way to populate the network. This requirement can be implemented only if the Platform allows the registration and the management of new IoT Brokers. For this reason, this requirement is satisfied only in the Core of Siemens MindSphere and by all FIWARE Platforms for definition.
7. **Register External Brokers.** The platform must support the registration of IoT External Brokers. This means that the IoT Platform should be capable of registering IoT Devices/Services of the External Broker into the IoT Platform. Brokers can be single- or multi-tenant and to recover the IoT Devices data model managed by the Broker is the first step to perform their registration. In the case of External Broker, the endpoint URL and the service and/or service path specifications would be needed to subscribe. None of the commercial platforms considered provides a solution for registering External Brokers, and thus making automated registration of their devices.
8. **Discover IoT Devices on IoT Brokers.** The platform must be capable to abstract IoT Devices from their IoT Brokers and protocols. This is needed for their registration and for their **classification and search**, which is based on their position, nature, value types and units, etc. In other words, it should be possible to discover/search (subscribe, get, send data) to/from IoT Devices independently from their position/connection in the IoT Network. The process of discovery must be manageable in the sense that its execution time can be scheduled, and possible with brokers that support a process for device discovery. The result should consist of an automated or semi-automated registration process of IoT Devices.
9. **Easy management graphic interface to list and test IoT Brokers, and IoT Devices** and query them. For each IoT Device, it has to be possible to perform testing activities. As seen before, not all the above-mentioned Platforms manage the IoT Broker, unless they use a specific add-on. So, this requirement is satisfied by each of them only for the kind of Devices they put on the offer.
10. **Manage IoT Device Model and Device Data Type ownership and access grant**. This permits assignment/changing of the ownership and the creation of access grants to the entities (Brokers, Devices, Models, etc.). In delegation management, it must be possible to list them (check the grants provided) and revoke the delegations. According to GDPR, any entity must start as private of the owner. The delegation should be possible for organizations, groups of users, and single users.

On other aspects, surveys about IoT Platforms are provided in [14], [15], [16].

III. The Role of IoT Directory in Snap4City Architecture

In order to enforce the above-described requirements into Snap4City IoT Platform, we have designed and developed a new concept that we have called IoT Directory. It extends the features of generic IoT Platforms with the management of (i) IoT Data Models, (ii) IoT External Brokers, (iii) discovery of IoT Devices of External Brokers, (iv) support the multi-tenancy, (v) support several organizations, (vi) GDPR compliance, etc.

In **Fig.1(left),** the registration process of IoT Brokers (**Internal or External**) in the Snap4City Platform is reported, where the organization is denoted by the subscription in the

Broker name. At the Broker registration into the IoT Directory, a number of parameters are needed including: the end point, security, name, External/Internal, single/ multiple-tenant, etc. The most important difference for Internal/external Brokers consists in the IoT Device management as explained in the following. The broker to be usable has to be granted to each specific user or public [13]. An user only belongs to a single organization for security and privacy aspects.

Once a Broker is registered, the IoT Directory automatically performs the subscription of the data platform to the new Broker for all its devices/topics, so that each new message that will be generated by the broker would be directly brokered to the data storage. In **Fig.1(right)**, the subscriptions are denoted by dashed lines, while the registrations are shown as solid lines. In most of the IoT Platforms, the storage is called Data Shadow, and allows to create the historical data of the IoT Devices. In Snap4City, the data storage feeding is performed by Apache NiFi, so that, the IoT Directory automatically performs the association between NiFi and the topics of the new **Internal or External** Broker. This is due to the necessity of having a robust, scalable tool with low latency to handle huge volume of data entering on the storage and coming from several devices and brokers. In **Fig.1(right)**, the processes of **IoT Device registrations** are depicted in both cases. In the case of **Internal Brokers,** the IoT Device registration is performed on the IoT Directory. The user may select the IoT Broker among those of the Organization and set a number of details. The registration may start with the exploitation of an IoT Device Model, the device ID, and then with the definition of GPS location, and all instance details. In Snap4City, the process can be performed: (i) on the IoT Directory via the user interface exploiting a model or not, (ii) exploiting API of the IoT Directory, (iii) using MicroServices in Node-RED which are based on the same API of (ii), (iv) using a set of automated registration processes starting from Excel Files/tables. Each registered IoT Device is registered on the **Knowledge Base**, KB, (implemented with Virtuoso on the basis of Km4City Ontology [17] for the semantic relationships and with Open Search for the time Series data) with all its information and metadata (static information). The KB management allows to index the device and establish all the relationships with the other city entities located in the same area, place, city, region, road, GPS position, etc. This information would be very useful when new messages arrive from a IoT Device in the storage via NiFi (which is represented by the ServiceMap in **Fig.1(right**), with **Smart City API** for providing access to the storage) they are connected to the right IoT Device description and relationships. The correct and complete indexing and **Smart City API** are fundamental to enable the exploitation of IoT data by IoT Applications (Node-RED microservices [14]) and Dashboards [16].

Finally, **Fig.2** shows the data flow during the usage, it illustrates the event-driven data flows. There are four ways for generating new IoT Messages, from:

- **IoT Devices** pass from a Broker and are passed to: (a) the NiFi, thus reaching the KB and storage, becoming part of historical data which can be accessed and queried from IoT App, Data Analytic and Dashboards; (b) Kafka to directly reach subscribed Dashboards via WebSockets, and IoT App.
- **IoT Apps** may be sent to IoT Broker or to Kafka front end broker. Thus, the message can reach: IoT Device for acting on it, storage, IoT App, Dashboards, etc. If a message is sent by a sensor-actuator (Internal or External), his Broker broadcasts it to Ni-Fi, which spreads it in turn. The IoT Apps are also used for massive registration of IoT Devices, and to perform data adaptation, ETL/ELT (Extract Transform/Load) processes.
- **Dashboards** are passed via Kafka toward the IoT App, or to an Internal Broker or direly into the storage. These messages can be regarded as Virtual IoT Devices to act on some sensors/actuators IoT Device, or even to simulate it. The produce messages may be sent to Internal/External Brokers, and so on.
- **IoT Directory** may generate a new message towards an IoT Broker (and may also read the last message from the broker). The generation of messages from the IoT Directory is typically used to check if the Internal Broker is alive and works correctly, and if the IoT Device messages are accepted.

  In the case of **registration of IoT Devices on an External Broker**, the Broker is not managed by the IoT Directory, and thus the Devices are registered in the External Broker by the third-party gateway manager without informing the IoT Directory. Thus, the IoT Directory need to recover the information needed for registering and indexing the devices into the KB. Then, the IoT Directory queries/harvest the external broker to get the structures of the IoT Devices (also called Discovery has to be started). The harvesting process may start having the broker API end point and also the service path in the case of the multi-tenancy broker. The harvesting has to be performed at the broker registration and every time a new Device is added, thus a periodic Discovery is needed. For the harvesting, the IoT Directory recognizes the Data Model and Data Types of the attributes to register them in KB in proper manner, to validate them. In the following subsections, the registration of IoT Devices on Internal and External Brokers are discussed. Please note that the registration of devices from External Brokers is one of the innovative aspects addressed by the IoT Directory which is capable to harvest the brokers and resolving semantic gaps on IoT device attributes/variables, see **Section 3.2.**

A. Registration of IoT Devices on Internal Brokers

The IOT Directory is fundamental for the definition and registration of IoT Device Models, i.e., data models. Focusing on the Internal IOT Brokers, the platform provides three ways to deploy IoT Devices:

- *Manual process:* the user can register an IoT Device by using a graphic interface, to input information. It is possible to register a device on the basis of a specific IoT Device Model, and to refer at a specific IoT Broker.

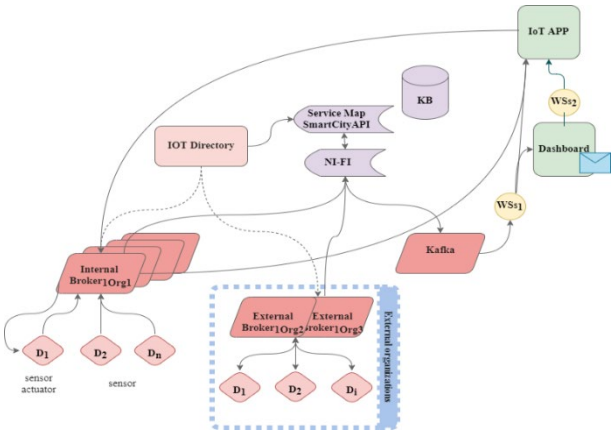

Figure 2. IoT Messages exchanged among entities: continuous lines are data flows, dashed lines indicate the tests that the user can perform to verify the IoT Devices/Brokers.

- *Bulk process:* the user can upload a file with a list of Devices, defining the IoT Broker, Model and Edge.
  - *IoT App process:* The user can build an IoT APP using Node-RED on which specific nodes can be used. The nodes accept JSON with parameters related to the chosen Device Model to register new Devices (see **Fig.4**).

  The registered IoT Devices are shown in a table in which the users can manipulate only those he/she created, no matter of the generation process. The users can also see in the list the public devices of the same organization, while the general administrator has a full visibility of all devices of all the organizations.

B. *Discovering & Registering IoT Devices from External Brokers*

The Snap4City Platform allows the registration of External Brokers using the broker URL or the couple URL and service identifier in the case of multi-tenancy. After the registration of an External Broker, the user can start the harvesting process and choose the time period for the update. It is important to highlight that Snap4City Platform may know a set of IoT Device Model (data models). Thus, in the best case, when the harvesting starts, it can recognize the device and its model (message format and device ID).

If the IoT Directory does not recognize the Device, the Device has to be Registered. To this end the IoT Directory can query the Broker to have more information to register it. On the other hand, the IoT Device may be compliant with a Data Model or not. If it is compliant the registration is straight forward. It is not compliant, each single attribute has to be recognized in terms of Value Type, Value Unit and Data Type (e.g., Temperature, Celsius, Float). The list of recognized/registered and non recognized/non registered Devices is presented. Through this interface, the user can resolve the problems manually defining the missing data and enabling the registration.

The most common harvesting (automated registration process) problem are due to the lack of matching with known attributes. The IoT Platform tries to identify the attribute kind in terms of value type, value unit and data type by performing query on data Dictionary and Km4City Ontology (via the connection from IoT Directory and ServiceMap by using Smart City API, and Dictionary API, **Fig.** 2. The recognition may have success in two cases: the data model is known but not this specific attribute or the model is not known, so all the attribute values of the Device are not recognized. In the first case is easy to fix the problem by applying a specific rule. In the second case, the Platform needs to learn a Rule for solving the attributes, thus defining a new Data Model in IoT Directory. Formally, the processing rule R is defined in EBNF as following:

R:= IF <condition list> THEN <action list>
<condition list>: = <c> | <c> AND <condition list>
<c> := <variable> <op> <constant>
<variable> := "device name" | "context broker" | "device type" | "model" | "value name"
<op> := "is equal" | "is not equal" | "Is null" | "contains"
<constant> := integer | float | string | list
<action list> := <a>| <a>, <action list>
<a> :=<action variable>: <action constant>
<action variable>:= "Data type" | "Value type" | "Value unit" | "Editable" | <Coded Healthiness criteria> | <Healthiness value>
<action constant>:= string

The rule is divided into two parts: If statement and then statement. In the first part, the user can define the condition that describes a set of devices, e.g., a device name in common. The *<op>* defines the operators, two of them ("is/ is not equal") can apply only on the number constant, others ("Is null" or "contains") only on the string constant. In the second part, the user can establish the action that makes devices or values valid. In other words, for the subset of Devices identify by the <condition list>, the *<action variable>* is modified as defined by *<action constant>*. An example of Rule can be:

IF "context broker" is equal "CONTEXTBROKER"
AND "value name" is equal "activePower"
THEN "value type":"power", "value unit":"W", "data type":"float"

In this case, the rule's builder selects a subset of invalid attributes named activePower, which have the Device subscripts a Broker named CONTEXBROKER which allows to manage the multi-tenancy aspects. Then, it changes the value type, the value unit and the data type in power, W and float.

In the IoT Directory, it is possible to search and edit the saved rules. When the user saves a rule, must choose the broker to which is applied. It is also possible to define a specific subset of service or service and service path to cope with multiservice brokers. Thus, the application of a rule is associated to each specific Broker or Organization since a Rule can be suitable for an organization and not functional for the others. Figure 5 illustrates the form of the rule builder.

IV. Validation Experiments

As above discussed, the harvesting of External Brokers may take into account one or more rules to recognize the attributes and data models, and thus to indexing the IoT devices in the right manner, and shortening time to registration,

dynamically add new IoT Devices registered on the Broker, thus reducing the gap from using Internal and External Brokers. The following validation values are calculated doing ad hoc experiments, which are repeated ten times. Focusing on the timing of the various processes, the user spends around 1 minute and a half on average to fill the form to add a new device with 10 attributes, and the system spends around 3 seconds to register the device. Meanwhile, if the user builds a device from a model, these timings are less than 1 minute on average to fill the form and around 1,35 seconds on average to submit the new device. Furthermore, if the user records a new device through IOT App, the system registers it in 623 milliseconds on average.

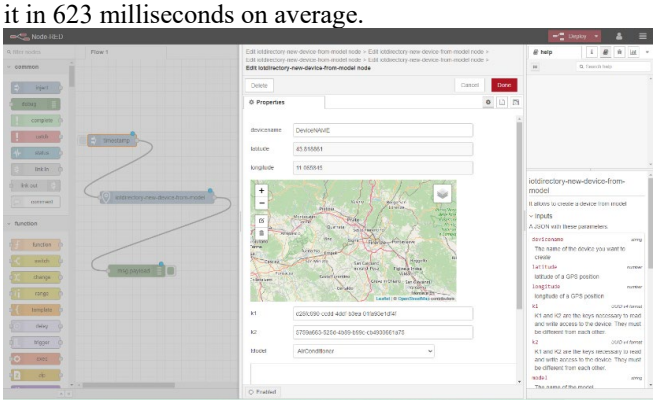

Figure 3. User interface in Node-RED for the creation of the device by IoT App is shown.

Focusing on the registration timing and considering an external multi-tenant broker with 37000 devices, the harvesting time is 25 minutes and 50 seconds on average. Meanwhile, the process's timing of the attributes of a specific FIWARE data model (Streetlight) ingestion is 37.1406 milliseconds on average, which results in the addition of 432 new Streetlight devices automatically. Of course, the user can make changes to IoT Devices structure after their automated or manual registration.

V. Conclusions

The proliferation of the IOT devices, brokers, networks, data models, operators and tenant, make the harmonization and management for IoT Platform a hard goal. This paper offers an analysis and a comparison among relevant existing platforms and delineates the basics requirements to achieve these aims. These identified requirements are in most cases not addressed by major platform which prefer to stay on their own end-to-end solutions with limited interoperability and capacity of exploiting the legacy IoT networks in place. The interoperable management of complex network has to pass from the IoT device registration which is typically a recurrent operation since the IoT networks are in continuous evolution. In this paper, the above-mentioned problems have been addressed introducing our concept of IoT Directory and reasoning tools to (i) manage Internal and External brokers, (ii) perform the automated registration by harvesting and reasoning of devices managed into external brokers single- or multi-tenant services, (iii) perform the automated registration and management of Data Models, and any custom Data Model. The solution has been developed and tested into Snap4City, an 100% open source IoT platform for Smart Cities and Industry 4.0, official FIWARE platform, EOSC, and lib of Node-RED. Thus, the resulting platform is more flexible than the others considered (Google IOT Cloud, MS Azure, AWS, Siemens Mindshare and IBM Watson). Furthermore, the proposed solution is also compliant with Smart Data Model of FIWIRE. The semantic interoperability of the platform can be improved by automatic generation of rules and completing the automation of the ingestion process. Furthermore, the process is helped by Km4City ontology and Data Dictionary to recognize the new or model data's semantic domain. The specific IoT Directory has been developed in the context of Herit-Data Project, the results have been validated in wide condition of the whole Snap4City network of more than 18 tenant, and billions of data.

Acknowledgment

The authors would like to thank the MIUR, the University of Florence and the companies involved for co-founding the Herit-Data project. Km4City and Snap4City (https://www.snap4city.org) are open technologies and research of DISIT Lab. Sii-Mobility is grounded and has contributed to the Km4City open solutions.

References

[1] Ghasempour, Alireza. "Internet of things in smart grid: Architecture, applications, services, key technologies, and challenges." Inventions 4.1 (2019): 22.

[2] Mekki, Kais, et al. "Overview of cellular LPWAN technologies for IoTdeployment: Sigfox, LoRaWAN, and NB-IoT." 2018 ieee international conferenceon pervasive computing and communications workshops (percom workshops).IEEE, 2018.

[3] Yousefpour, Ashkan, et al. "All one needs to know about fog computing andrelated edge computing paradigms: A complete survey." Journal of SystemsArchitecture 98 (2019): 289-330.

[4] Adelantado, Ferran, et al. "Understanding the limits of LoRaWAN." IEEECommunications magazine 55.9 (2017): 34-40.

[5] https://aws.amazon.com/iot

[6] https://siemens.mindsphere.io/en

[7] https://cloud.google.com/solutions/io

[8] https://azure.microsoft.com/en-us/overview/iot

[9] https://ww.ibm.com/watson

[10] https://www.snap4city.org

[11] Cheng B., G. Solmaz, et al.,"FogFlow: Easy Programming of IoT Services Over Cloud and Edges for SmartCities," in IEEE Internet of Things Journal, vol. 5, no. 2, pp. 696-707, April 2018,doi: 10.1109/JIOT.2017.2747214.

[12] Badii C., P. Bellini, A. Difino, P. Nesi, G. Pantaleo, M. Paolucci, "MicroServicesSuite for Smart City Applications", Sensors, MDPI, 2019. https://doi.org/10.3390/s19214798

[13] Badii C:, P. Bellini, A. Difino, P. Nesi, "Smart City IoT Platform RespectingGDPR Privacy and Security Aspects", IEEE Access, 2020.10.1109/ACCESS.2020.2968741

[14] Ammar, Mahmoud, et all. "Internet of Things: Asurvey on the security of IoT frameworks." Journal of Information Security andApplications 38 (2018): 8-27.

[15] Ray, Partha Pratim. "A survey of IoT cloud platforms." Future Computing andInformatics Journal 1.1-2 (2016): 35-46.

[16] Bellini P., D. Cenni, et al, "Smart CityControl Room Dashboards: Big Data Infrastructure, from data to decisionsupport", Journal of Visual Languages and Computing, 10.18293/VLSS2018-030

[17] Bellini P., D. Nesi, et al, "Federation of Smart City Services viaAPIs", Proc of 6th IEEE International Workshop on Sensors and Smart Cities, with IEEE SmartComp, 14-17 Sept. 2020, Bologna, Italy. http://ssc2020.unime.it/