An Architecture for Interoperable IoT Ecosystems

Stefan Schmid[1], Arne Bröring[2], Denis Kramer[3], Sebastian Käbisch[2], Achille Zappa[4], Martin Lorenz[5], Yong Wang[6], Andreas Rausch[6], Luca Gioppo[7]

[1] Robert Bosch GmbH,
Stuttgart, Germany
stefan.schmid@bosch.com

[2] Siemens AG,
Munich, Germany
{arne.broering | sebastian.kaebisch}@siemens.com

[3] Bosch Software Innovations GmbH,
Stuttgart, Germany
denis.kramer@bosch-si.com

[4] Insight Centre for Data Analytics, NUI Galway,
Galway, Ireland
achille.zappa@insight-centre.org

[5] Atos IT Solutions & Services GmbH,
Vienna, Austria
martin.lorenz@atos.net

[6] Technical University Clausthal,
Clausthal-Zellerfeld, Germany
{yong.wang | andreas.rausch}@tu-clausthal.de

[7] CSI-Piemonte,
Torino, Italy
luca.gioppo@csi.it

Abstract. The Internet of Things (IoT) is maturing and more and more IoT platforms that give access to *things* are emerging. However, the real potential of the IoT lies in growing IoT cross-domain ecosystems on top of these platforms that will deliver new, unanticipated value added applications and services. We identified two crucial aspects that are important to grow an IoT ecosystem: (i) interoperability to enable cross-platform and even cross-domain application developments on top of IoT platforms as well as (ii) marketplaces to share and monetize IoT resources. Having these two crucial pillars of an IoT ecosystem in mind, we present in this article the *BIG IoT architecture* as the foundation to establish IoT ecosystems. The architecture fulfills essential requirements that have been assessed among industry and research organizations as part of the BIG IoT project. We demonstrate a first proof-of-concept implementation in the context of an exemplary smart cities scenario.

Keywords: Internet of Things, Architecture, Interoperability, Marketplace

1 Introduction

The idea of the Internet of Things (IoT) [1] has become more and more a commercial reality that spans various application domains, from smart homes, over smarter cities, to Industry 4.0. Various IoT platforms are upcoming: Cloud–level platforms such as Evrythng[1] or ThingWorx[2], and also on premise solutions such as Bosch's IoT Suite[3]. However, up to now, these IoT platforms failed to form vibrant IoT ecosystems. This is due to the large number of stakeholders, including developers and providers of platforms, services and applications. They require *marketplaces* that enable the monetization of their IoT resources. Once such marketplaces are established, revenue streams can be shared across all contributing stakeholders. A crucial task of a marketplace is to provide functionalities for advertising, discovery and orchestration of IoT services to facilitate their usage.

However, before such marketplaces can bring their effect, a serious market barrier needs to be tackled: the missing *interoperability*. A recent McKinsey study [2] estimates that a 40% share of the potential economic value of the IoT directly depends on interoperability between IoT platforms. Today, we are dealing with various vertically oriented and mostly closed systems. Architectures for IoT are built on heterogeneous standards (e.g., OMA LWM2M [3], OGC SWE [4], or OneM2M [5]) or even proprietary interfaces. This causes interoperability problems when overarching, cross-platform and cross-domain applications are to be built. Additionally, it leads to barriers for small innovative business, which cannot afford to offer their solution across multiple platforms.

In order to address these shortcomings in today's IoT landscape, this article concretizes our vision presented in [6]: It presents the *BIG IoT architecture* as enabler for establishing IoT ecosystems. It overcomes the above described hurdles through (1) a common Web API, (2) semantic descriptions of resources and services, as well as (3) a marketplace as a nucleus of the ecosystem. We implement this architecture as part of the BIG IoT project[4]. This will allow new applications, e.g., by combining data from multiple platforms (e.g., parking information from various smart city platforms). In addition, platforms from multiple domains (e.g. home and city) and regions will be combined, such that applications can utilize all relevant information and work seamlessly across regions.

To ignite an IoT ecosystem based on the developed concepts, the BIG IoT project involves overall 8 IoT platforms. There are 6 cloud- or infrastructure-level platforms: Bosch's Smart City platform, based on the Bosch IoT Suite[3], CSI's Smart Data[5] platform, OpenIoT [7], Vodafone's Mobile Analytics Platform, VMZ's TIC[6] platform, and WorldSensing[7]. Further, there are 2 device-level platforms: Bosch's BEZIRK[8] platform and Econais' Wubby[9] platform. Together with multiple service

[1] http://www.evrythng.com
[2] http://www.thingworx.com
[3] https://www.bosch-si.com/products/bosch-iot-suite/platform-as-service/paas.html
[4] http://big-iot.eu
[5] http://www.smartdatanet.it
[6] https://viz.berlin.de/en/home
[7] http://www.worldsensing.com
[8] http://www.bezirk.com/platform.html

and application implementations, these platforms will be BIG IoT-enabled and evaluated in 3 different pilots (Barcelona, Berlin/Wolfsburg, and Piedmont).

The remainder of this article is structured as follows. Section 2 presents related work and outlines an overview of different research projects in this field. Section 3 outlines the high-level concepts and requirements for IoT ecosystems. Section 4 describes the BIG IoT realization of such an IoT ecosystem architecture, which is then demonstrated in a proof-of-concept in Section 5. Finally, we conclude this article in Section 6.

2 Background & Related Work

Various related works exist that contributed to the advancement of IoT architectures design. Most related to our work are other large research projects in context of the IoT. This section lists some of such approaches to give an overview of the research field and highlights the unique approach of our work in BIG IoT.

A prominent project in this context is the Internet of Things Architecture (IoT-A) project [8], which developed a comprehensive architectural reference model as a foundation for interoperability of IoT systems, including guidelines for the design of protocols and interfaces. However, other than IoT-A, which can be used as a blueprint for the development of an IoT platform, this work develops an architecture that focuses on integrating existing systems, components, and stakeholders of the IoT.

Another lighthouse IoT framework project is FI-WARE [9]. It develops a framework of so-called generic enablers to support IoT developments. Our approach differs from the FI-WARE idea, as we do not aim at creating another unified platform or platform building blocks, but enabling the coexistence and distributed collaboration of existing and already commercially deployed platforms to foster an easy creation of portable services by third party providers.

A Semantic Web-based design of a middleware platform for the IoT has been developed in the OpenIoT project [7]. While OpenIoT assumes the use of a single sensor middleware platform and its integration within a common cloud computing infrastructure, it does not address cross-platform mechanisms. This is however a focal topic of the work described in this article. In fact, the OpenIoT platform is integrated into the BIG IoT project as one IoT platform of the overall ecosystem.

VITAL [10] aims at virtualized filtering and complex event processing mechanisms over a variety of IoT architectures. It focuses on an abstract virtualized digital layer that will operate across multiple IoT architectures. In that sense, VITAL has similar goals of integrating different IoT platforms. However, it is a domain specific effort, by restricting itself to smart cities. The project develops a centralized operating system, called Vital-OS, which manages and monitors all systems and data. In contrast, our work follows a domain-agnostic approach that generalizes emerging platforms and enables semantic interoperability to provide unified APIs.

The objective of CityPulse [11] has been to develop a distributed framework for the semantic discovery and processing of large-scale real-time IoT data and relevant social data streams for knowledge extraction in a city environment. CityPulse focuses

[9] http://www.wubby.io/

on developments for the application layer. Their services could be integrated and run on top of the common API designed by BIG IoT.

The IoT@Work project [12] was a deep dive into the industry automation domain with its very specific requirements. The approach presented a novel solution for flexible production. The BIG IoT project in contrast aims at cross-domain applications and making use of existing platforms and installations in a more generic sense. By fostering the emergence of open ecosystems, our approach diverts from the specific one of IoT@Work.

One of the intentions of the BIG IoT project is to bring its approaches to standardization in order to reach an interoperable IoT platform landscape. In that sense, BIG IoT members are involved in the W3C Web of Things (WoT) [13] activities, which is going to be standardized in parallel to the BIG IoT project. The W3C WoT group was founded in Spring 2015 and its major motivation for initiating this group was also based on the fact that the IoT suffers from a lack of interoperability across platforms. BIG IoT members are mainly involved in the topic Thing Description. Development and experiences in WoT and BIG IoT are regularly synchronized in order to learn and benefit from each other.

Mineraud et al. [14] analyze the technological gaps of today's IoT platforms. Specifically, they highlight the fact that data and device catalogs as well as billing of consumers of the IoT data sources is generally missing. They suggest that marketplace functionality needs to be provided to enable the full potential of IoT platforms. This recent study shows that our work in the BIG IoT project is highly relevant. By building up on the previous works outlined above, we focus on reusing existing technologies with the goal of igniting IoT ecosystems.

3 High-level Architecture Concepts and Requirements

This section first defines the terminology and key concepts for an IoT ecosystem architecture, then it identifies requirements from our stakeholders, deducts architectural implications, and high-level design decisions that influence and guide the design of the concrete BIG IoT architecture in Section 4.

3.1 Terminology and Conceptual Model for an IoT Ecosystem

Figure 1 defines the generic concepts that we identified within an IoT ecosystem and the interactions between them. The core concepts are: *offerings*, *(offering) providers* and *consumers*, and the interactions of *registering* and *discovering* offerings via a marketplace, and *accessing* the resources offered by a provider.

An *offering* encompasses a set of IoT resources, typically a set of related information (e.g. low-level sensor data or aggregate information) or functions (e.g. actuation tasks or computational functions), that are offered on a marketplace.

Providers register their offerings on a marketplace and provides access to the offered resources via a common API. A provider can be either a platform or a service instance that offers available resources, i.e., some information or access to functions that it wants to share or trade on the marketplace (e.g. an IoT platform of a parking lot provider). *Consumers* discover and subscribe to offerings of interest via a marketplace

in order to access the resources. A consumer can be either an application or service instance that requires access to IoT resources in order to implement an intended service or function (e.g., a smart parking service provided by the city).

In technical terms, a provider registers its offerings on the marketplace by providing an *offering description* for each offering. An offering can for example entail parking information for a city and include data such as geo location or address of the parking lot, the type of lot (e.g. garage or on-street), available spots, occupied spots, etc. In general, to increase interoperability between different IoT platforms, the offering description should be provided in a machine interpretable manner, e.g., based on RDF [15] models. All relevant communication metadata should be provided on how the offering can be accessed (e.g., endpoint URL, which HTTP method, etc.). As a default vocabulary set, the offering description should include a local identifier (unique to a provider), a name of the offering, and the input and/or output data provided to a consumer when the offering is accessed. The description may also include information about the region (e.g. the city or spatial extent) where the resources relate to, the price for accessing the resources, the license of the data provided, the access control list, etc.

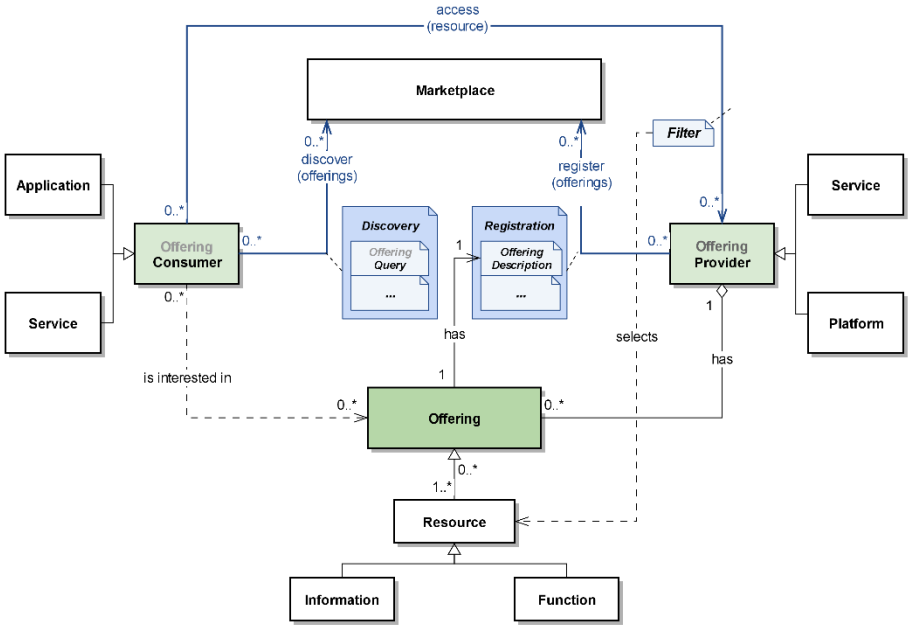

Figure 1. Conceptual model for an IoT ecosystem.

Consumers discover offerings of interest on the marketplace by providing an *(offering) query*. The query entails a specification of the type of offerings the consumer is interested in. For example, a consumer can provide a description of the desired resources (such as type of parking information), and also define the maximum price, the desired license types, the region, etc. Upon a discovery request, the marketplace identifies all matching offerings and returns them to the consumer. The consumer can then choose the offerings of interest and subscribe to those on the

marketplace. To refine the data to be requested, a consumer can also provide a *filter* in the access request, which is then applied at the provider end to filter the resources to be delivered.

3.2 Use Cases and Requirements

The high-level requirements for designing the architecture have been identified through discussion of relevant use cases and from a qualitative survey among the stakeholders from industry and research involved in the BIG IoT project. Clusters of requirements have been identified, as described in the following.

1.) Core technology – Given the overall goals of our work, namely to facilitate IoT ecosystem creation and to enable resource providers to trade and monetize their IoT resources, and consumers to discover and utilize them across platform and domain boundaries, we have identified crucial high-level functional requirements: First, IoT platforms and services need to be able to offer and register IoT resources on a marketplace and provide easy access to the resources via a common API. Second, applications and services shall be able to discover desired IoT resources via a marketplace and access them across heterogeneous platforms or services via a common API. Third, resource providers shall be able to monetize their assets (information and functions) via a marketplace. Fourth, resource consumers shall be able to discover new resource providers at run-time and leverage their resources immediately.

In conclusion, we identified three technological pillars that are key for the development of an IoT ecosystem: a centralized marketplace, common API(s), and a software development kit (SDK) for easy integration with the ecosystem. The API and its implementation, the SDK, need to be developed in an open source/community process.

2.) Developer support – In order to grow an IoT ecosystem, it is crucial to lower the hurdle of joining the ecosystem, and thus, support developers in the process of extending their IoT platforms, services or applications. These scenarios involve developers that a) extend their platform to support the common API and offer resources to a central marketplace, and b) develop a service or application, which uses the common API to gain access to the marketplace to discover offerings and connect to their provider platforms or services. In this context, we identified three essential use cases. First, a developer studies the BIG IoT documentation, example code and downloads the SDK. Second, a service/platform developer implements a service or extends an IoT platform to register a resource offering on the marketplace. Third, an application/service developer implements an application/service, which utilizes a resource offering discovered via a marketplace.

3.) Exchange of resource offerings – This cluster of use cases defines how a) providers can offer their resources on a marketplace, and how b) consumers can search for offerings and access them. The derived requirements are: First, a service/platform registers a resource offering on a marketplace. Second, a service/application discovers offerings via a marketplace and accesses them on the platforms/services.

4.) Charging and billing – One of the core functionalities of an IoT ecosystem marketplace is to enable providers to monetize the access and use of their resources.

Therefore, the following two requirements describe the collection of accounting and usage data, as well as further functions necessary for charging and billing. First, platform/service/application instances perform accounting of the accessed resources. Second, a marketplace offers accounting and charging information to the involved stakeholders.

5.) Non-functional requirements – First, the integration of existing and new IoT applications, services, and platforms with a marketplace shall be low-effort. Second, the common API and the marketplace implementations shall be highly scalable to support large-scale IoT deployments. Third, the communications and interactions among consumers, providers, and the marketplace shall be secure, as this is a crucial aspect for any IoT deployment to work.

3.3 Platform Integration Modes

For the integration of heterogeneous IoT platforms into IoT ecosystems, we have analyzed the needs and constraints of the 8 platforms involved by the BIG IoT project partners (see Section 1). The following challenges have been identified:

1. The implementation of the API for interaction with the marketplace, and to offer access to consumers must be low effort.
2. Platform providers that use off-the-shelf platform solutions, and thus have no access to the source code of their platform, need alternative means to integrate their platforms into an ecosystem.
3. Constrained[10] device-level platform providers need infrastructure-level support to overcome the availability and cost limitations of such platforms.

In order to address challenge 1, BIG IoT provides developers an *SDK* comprising a library for using the API. This way, a developer can extend an IoT platform programmatically by means of an easy-to-use programming interface. While we currently focus on Java, the SDK will be provided for common programming languages and development environments.

To cope with challenge 2, we suggest that affected platform providers develop a *gateway-service*. Such a gateway-service sits between the existing platform, and the marketplace or consumer applications/services. We envision that open source gateway-service implementations will become available for common IoT platform types.

In order to deal with challenge 3, we support affected platform providers by releasing an open source *proxy-service* implementation together with an extended SDK that allows easy integration of such constrained device-level platforms with the proxy-service and the marketplace. The main functionality of the proxy-service is to store informational resources that are offered by the device-level platform and serve them to interested consumers upon request. With respect to tasks or actions that need to be processed by such device-level platforms, the proxy-service queues them until

[10] *Constrained* in this context means that the platforms may not always be accessible (either due to energy saving reasons or with wireless coverage) and/or their backhaul connection might incur costs based on a "pay-per-use" plan (e.g. mobile phones or battery-powered sensors).

the platform connects and pulls the received tasks or actions. The response of a task or action is also proxied by such a service.

We validated the different options with all the platform providers involved in BIG IoT. The results show that 5 out of 8 platform providers are interested in the API library to extend their platform programmatically. In addition, 5 out of 8 providers indicated interest in the gateway-service based integration option. From the 2 device-level platform providers involved in the project, both confirmed interest in the proxy-service.

3.4 High-level Design Decisions

This section draws high-level design decisions for the architecture work based on the surveyed needs of the BIG IoT platform providers and the considerations discussed above.

1. Focus the marketplace functionality on an IoT resource exchange.

The functional scope of a marketplace in an IoT ecosystem can be broad. We evaluated the following possible key functional options:

- Resource exchange – for IoT resource providers and consumers to publish and discover their resource offerings and facilitate the resource exchange;
- Application or service store – for IoT developers to trade their applications or services software; and
- Hosting environment – for application, service or platform providers to host their run-time systems.

Based on a survey among the BIG IoT partners, we identified the resource exchange functionality as most crucial, and focus the BIG IoT marketplace on this. Nevertheless, the marketplace may be extended towards other functionalities in the future.

2. Consumers access IoT resources directly on the provider.

For scalability reasons and to keep IoT resources under full control of the providers, we propose not to store IoT data on a marketplace, but to enable easy access to the resources directly on the provider end. This design decision has the advantage that the marketplace only requires the ability to publish and discover the resource offerings (i.e., the descriptions of the resources), and to facilitate the direct access (e.g., through authentication of consumers and accounting support), but the actual resources remain stored and managed on the provider infrastructure.

3. Providers and consumers can participate on multiple marketplaces.

In order to avoid a marketplace lock-in, we propose to allow providers and consumers to use and interact with multiple marketplace instances at the same time. The advantage is that providers can offer their resources on multiple marketplaces, and thus, minimize the risk of integrating the API without good prospects to regain the initial investment of joining the ecosystem or running the risk of a vendor lock-in. Likewise, consumers can participate on multiple marketplaces.

4 The BIG IoT Architecture

This section describes the BIG IoT architecture as a realization of the generic concepts and requirements for IoT ecosystems presented in Section 3. A first implementation by the BIG IoT project is currently in progress. As shown in Figure 2, we distinguish the following 5 core building blocks:

1.) BIG IoT enabled platform – this IoT platform implements (as a *provider*) the common API, which is called the *BIG IoT API*, to register offerings on a BIG IoT Marketplace, and grants BIG IoT Services or Applications (as *consumers*) access to the offered resources.

2.) BIG IoT Application – this application software implements and uses the BIG IoT API, (as a *consumer*) to discover offerings on a BIG IoT Marketplace, and to access the resources provided by one or more BIG IoT Services or Platforms (as *providers*).

3.) BIG IoT Service – this IoT service implements and uses the BIG IoT API to register offerings on a BIG IoT Marketplace (as a *provider*) and/or to discover and access offerings provided via a BIG IoT Marketplace (as a *consumer*).

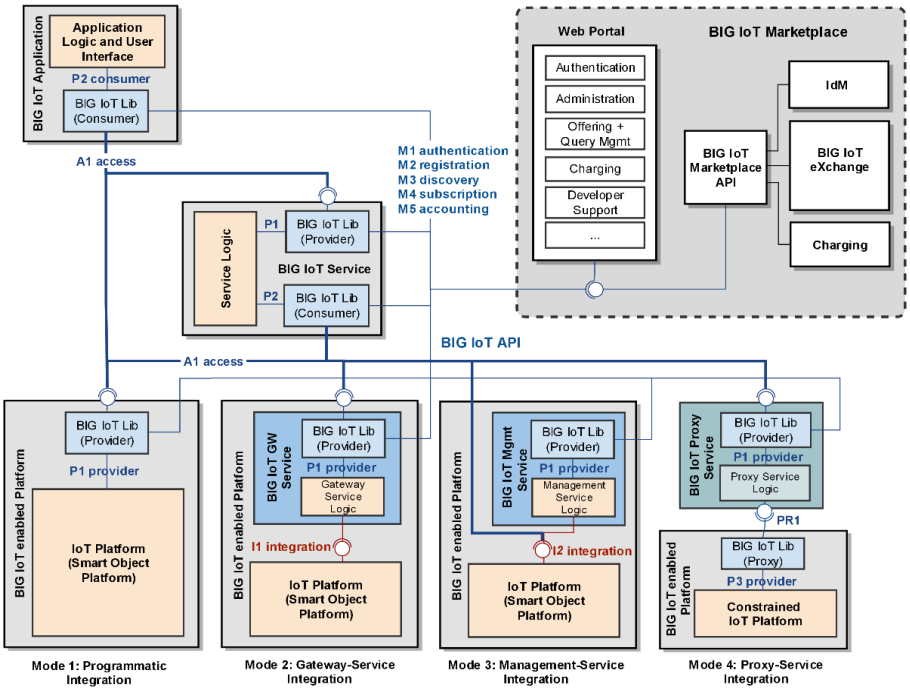

Figure 2. The design of the BIG IoT architecture.

4.) BIG IoT Marketplace – this composite system consists of sub-components:

The *Marketplace API* serves as an entry point for all communications and interactions with the marketplace; the *Identity Management Service (IdM)* which authenticates and authorizes providers and consumers; the *eXchange*, which allows registration and discovery of offerings using semantic technologies; the *Web Portal*

for users of the Marketplace; and the *Charging Service*, which collects accounting information. The Web Portal allows the users of a marketplace (typically organizations) to register and create accounts for their developers and administration personnel who in turn can create and register new provider or consumer instances, define new offerings and queries (for supported application domains), query and subscribe to offerings of interest, and manage those conveniently via a Web browser.

5.) BIG IoT Lib – this is an implementation of the *BIG IoT API* that supports service and application developers. The BIG IoT Lib consists of a *Provider Lib* and a *Consumer Lib* part. It translates function calls from the respective application or service logic, or the platform code into interactions with the marketplace, or peer-services or -platforms. The Provider Lib allows a platform or service to authenticate itself on a marketplace and to register offerings. As described in Section 3.1, the offering description is machine-readable and we base it on RDF [15] models. It incorporates the W3C WoT [13] *Thing Description* design pattern: offerings can be semantically described by integrating existing domain contexts (e.g., specific vocabularies for smart cities, smart home, or manufacturing). The Consumer Lib allows an application or service to authenticate itself on a marketplace, to discover available offerings based on semantic queries, and to subscribe to offerings of interest. The use of semantic technologies enables the eXchange to perform semantic matching even in case providers and consumers use different semantic models or formats, as long as a common meta-model defines the relations/mapping between the different semantic models and converters for the different semantic formats are supported.

4.1 Architecture Integration Modes

To comply with the requirements identified in Section 3.3, the architecture supports the following platform integration modes:

Mode 1: the platform developer uses the *Programming Interface P1* provided by the *Provider Lib* to programmatically extend an existing or new IoT platform.

Mode 2: the provider develops and operates a *BIG IoT Gateway Service*, which handles all BIG IoT related interactions and translates the relevant requests into calls supported by the existing platform (*Integration Interface I1*).

Mode 3: the provider develops and operates a *BIG IoT Management Service*, which handles the interactions with the marketplace. It integrates with the legacy platform by implementing the *Integration Interface I2*. Access to the resource offerings is provided directly by the legacy platform.

Mode 4: the provider develops and operates a *BIG IoT Proxy Service*, which handles the interactions with the marketplace and offers the *Access Interface A1*. The proxy-service acts as an "always-available" proxy on behalf of a typically constrained device-level platform.

4.2 Interfaces and Interactions

Besides the core components, Figure 2 also depicts the relevant interfaces of the architecture. The *Programming Interfaces P1* and *P2*, provided by the BIG IoT Lib, are offered to developers to connect their components with the marketplace and other

entities. For easy integration of constrained device-level platforms, a special BIG IoT Lib is provided, which allows developers to easily interact with the BIG IoT Proxy Service (via the *Programming Interface P3*).

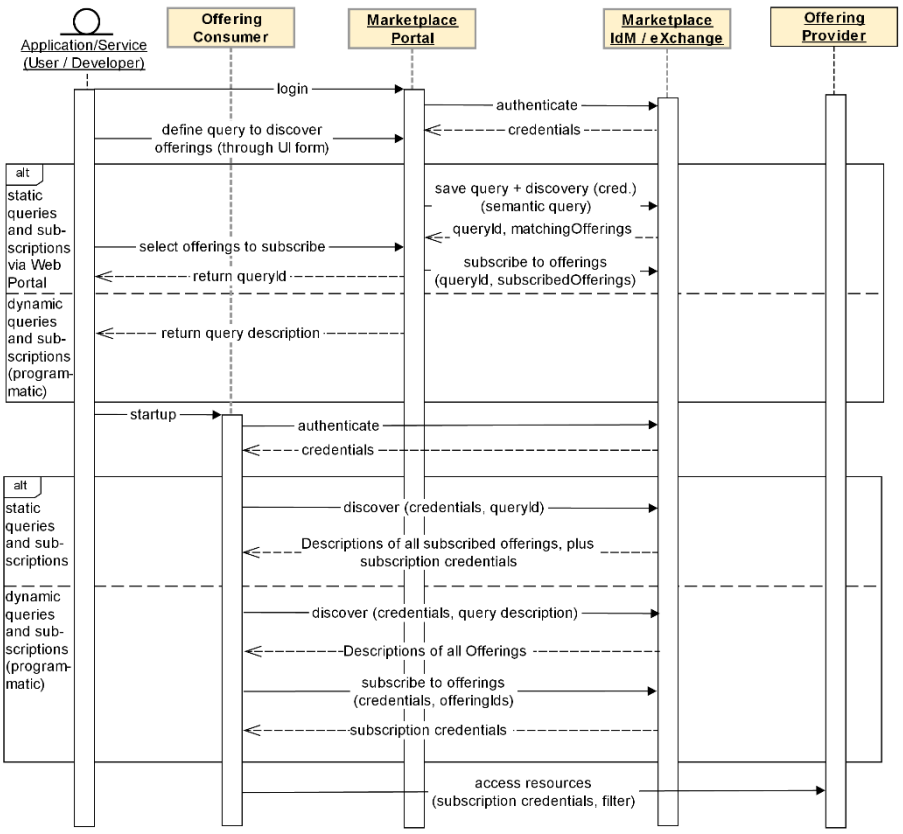

Figure 3. Sequence of offering discovery and subscription.

The BIG IoT Marketplace provides five interfaces to allow interactions with its services. The *M1 interface* is used by the provider and consumer instances to authenticate themselves on the marketplace at start-up. Upon successful authentication, the Provider or Consumer Libs will obtain the required credentials for any further communication and interaction with the marketplace. The *M2 interface* is used by providers to register/deregister offerings, while the *M3 interface* is used by consumers to discover offerings on the marketplace. Once a registration request is received, the BIG IoT eXchange validates the offerings and stores them in a semantic database. To subscribe/unsubscribe to offerings, consumer applications use the *M4 interface*. With a subscription, a consumer indicates its intent to access the offered resources, and confirms its consent with respect to the offering's license, price, etc. Once an offering is subscribed, the eXchange provides the consumer unique credentials to access this offering. In case the offering has expired or has been updated by the provider, the eXchange revokes the subscription and indicates the

cause in the response. The *M5 interface* is used by consumers and providers to send accounting information in regular time intervals to the Charging Service. Accounting types (e.g. per message) can differ between offerings, and are specified by a provider in the offering description. The interfaces M1-M4 are used in the same way by the Web Portal.

The *Access Interface A1* is the interface via which a consumer gets access to resources offered by a provider. Depending on the Provider Lib implementation, it will support different access means. All Provider Libs shall support the HTTP-based request/response access. Optionally, a Provider Lib can also support other protocols (e.g. WebSockets, MQTT) or other access paradigms (e.g. streaming).

Figure 3 describes the discovery (M3) and subscription (M4) to offerings on the marketplace in more detail. Once a query has been created by a developer via a Web UI, we distinguish between two modes: static and dynamic. In static mode, the developer or administrator of a consumer application or service selects and subscribes to the offerings of interest manually, via the Web portal. In dynamic mode, queries can be programmatically refined by the application or service logic, e.g. in order to consider information that is only available at run-time (e.g. location) and the subscriptions to offerings is automated based on consumer-defined policies. The dynamic mode is needed in case an application or service is designed to discover and integrate new data sources at run-time, e.g. in order to incorporate emerging offering providers automatically.

5 Proof-of-Concept Implementation and Demonstrator

This section presents a proof-of-concept implementation of the BIG IoT architecture components for a smart city use case. In an end-to-end scenario, the practicability of the BIG IoT architecture, including the marketplace and the API for an interoperable IoT ecosystem and the feasibility of the approach is demonstrated.

The overall goal of the developed architecture is to ease the interoperation of IoT platforms, services and applications despite technological and organizational boundaries. This scenario showcases that the run-time discovery and integration of IoT resources provided by heterogeneous platforms and various organizations becomes possible through the developed API and the marketplace. Although this scenario incorporates platforms from the smart city domain, the BIG IoT components and interfaces can be utilized in other domains as well. The key components are (1) the eXchange backend and the Web Portal of the BIG IoT Marketplace, (2) a demo Web application, (3) the cloud-level OpenIoT platform offering parking space data, and (4) the device-level Wubby platform offering air quality data. Figure 4 shows the key components as well as the interfaces and connections between those components. By default, the two platforms shown at the bottom offer their own proprietary interfaces. To integrate these platforms into the BIG IoT ecosystem, two gateway-services (shown in blue) are implemented according to integration mode 2 (Section 4.1). Those gateway-services implement the adaptation of the proprietary platform interfaces to the BIG IoT API. This adaptation is facilitated through usage of the BIG IoT Lib (shown in yellow). This library offers the access interface for consumers to access the resources, and can be used to interact with the marketplace as a client. The

library is also used by the demo Web application (shown in green). Using the BIG IoT Lib, simple method calls in the particular programming languages (here: Java), makes it easy to discover the relevant resource providers and to utilize the access interface of the heterogeneous platform.

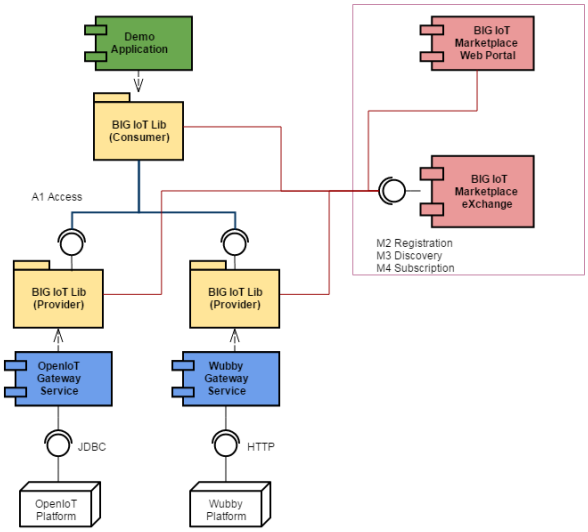

Figure 4. Components of the proof-of-concept demonstrator implementation.

The sequence of interactions in this demonstration is illustrated in the following. An application developer is implementing a Web application (Figure 6) that is supposed to visualize available parking spaces in smart cities. First, the developer visits the marketplace Web Portal and fills out the UI form accordingly to search for available resource offerings based on a semantic type that is of interest to her. Figure 5 shows the screenshot of the prototypical implementation of the marketplace Web Portal. We assume that at this point, only a few parking information offerings are found for this category. Nevertheless, the user is presented with a *Query ID* that is associated with the respective parameters. The user utilizes the returned *Query ID* and places it in her application code in order to allow her Web application to perform regular discovery requests for the offerings interested on the marketplace. Running the application triggers the discovery request based on the *Query ID*, however, only few parking spaces are shown on the map, as not many offerings of type "parking" are registered or active.

In a next step, a new user (platform provider) visits the marketplace portal to create an offering called "Barcelona Parking Sensors" and tags it with the same semantic type. After this creation, the offering is still inactive. The portal presents the provider with an *Offering ID*. This ID is used by the provider as a parameter in the OpenIoT gateway service. Once the provider starts the gateway service, it automatically registers the offering on the marketplace using the created *Offering ID* and marks it then as active.

Coming back to the Web application which makes periodic discovery requests based on the defined *Query ID*, it now finds the new offering (of the desired semantic type) and automatically accesses and integrates the data in the application. As a result,

the application visualizes the newly found parking spaces as markers at their specific locations (Figure 6).

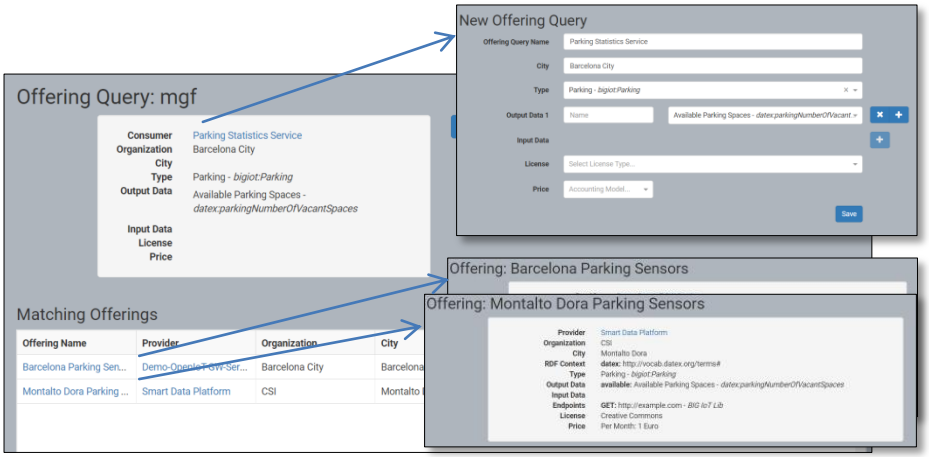

Figure 5. Screenshot of the marketplace portal UI to create and view queries.

Once the application receives a new offering from the marketplace, it checks all relevant information (e.g., price for accessing the offering, or license agreement) whether they meet the user's requirements. Then, the application subscribes to the matching offering, and eventually accesses the offering to retrieve the parking information. The access to the parking information on the provider platform is enabled by using a direct access interface provided by the BIG IoT Provider Lib. When the Web application calls the access method (provided by the BIG IoT Consumer Lib), the OpenIoT Gateway Service translates the requests for the parking information into a proprietary call to the OpenIoT platform and returns the data to the Web application.

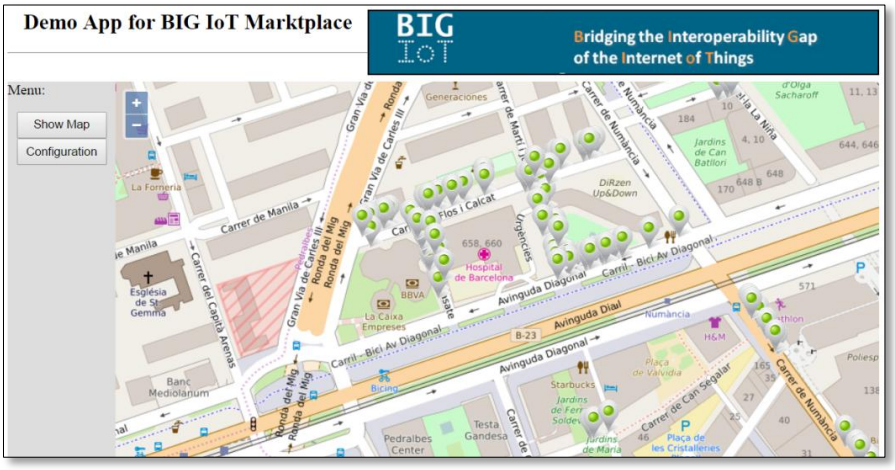

Figure 6. Screenshot of the Web application to view the available parking spots.

6 Conclusions & Outlook

Grounded in our vision of interoperable IoT ecosystems [6], we define in this article generic concepts for IoT ecosystem architectures, such as marketplaces, offerings, providers, consumers. Based on this core terminology, we present guiding use cases and requirements for the architecture, which were derived from surveys among the industrial and research partners of the BIG IoT project. We realize those generic considerations in the concrete *BIG IoT architecture*, by describing the key building blocks, their interfaces, and interactions. Finally, we present a first proof-of-concept implementation and demonstrator in order to illustrate the core architectural concepts, their feasibility, and the advantages of the architecture. While this scenario incorporates IoT platforms from the smart city domain, the BIG IoT components and interfaces are likewise applicable in other domains.

The demonstrator shows that the defined architecture is capable of: (1) solving the discovery challenge of available IoT resources for application and service providers, despite the fact that resources are collected and stored across heterogeneous platforms and systems, across large geographic spaces, and by a multitude of stakeholders and organizations, who are mostly not even aware of each other; (2) bridging the interoperability gap among heterogeneous IoT applications, services and platforms, which are using various standards and technologies, and operate on different scales (cloud-level vs. device-level platforms); and (3) addressing the evolvability problem of applications and services, who rely mostly on manual integration of continuously emerging IoT resource providers (e.g., new data sources), and thus, require growing development efforts to keep their applications or services up-to-date.

First, the BIG IoT architecture, with its marketplace, overcomes those challenges by introducing "places" for resource providers and consumers to meet and exchange their resource offerings and demands, and discover each other. Second, based on the BIG IoT API, the heterogeneous platforms and systems involved are able to access and exchange resources using standard protocols and frameworks. Finally, since the BIG IoT architecture supports the discovery of providers and their resources as well as the access to the resources at run-time, IoT applications and services are now able to integrate automatically emerging resource providers at run-time.

Key enablers for addressing the discovery challenge are semantic technologies. They facilitate the matching of resource offerings and queries across heterogeneous systems and diverse stakeholders, and also help to overcome the interoperability challenge. In the future, semantic vocabularies for specific application domains need to be established. This is needed in order to enable semantic matchmaking for IoT offering discovery on the marketplace. The BIG IoT project aims at using and extending existing and proven vocabularies, such as schema.org.

The detailed specification of the BIG IoT API, and in particular the use of semantic technologies to describe resource offerings, queries, the resources themselves, as well as the detailed specification of the BIG IoT Marketplace architecture, including the eXchange and the use of semantic databases, is ongoing work. To ground these specifications in public standards, we are actively contributing to the W3C Web of Things group and will continue doing so in the future.

Acknowledgments. This work is financially supported by the project “Bridging the Interoperability Gap” (BIG IoT) funded by the European Commission's Horizon 2020 research and innovation program under grant agreement No 688038.

References

[1] N. Gershenfeld, R. Krikorian and D. Cohen, "The Internet of Things.," *Scientific American,* vol. 291, pp. 76-81, 2004.

[2] J. Manyika, M. Chui, P. Bisson, J. Woetzel, R. Dobbs, J. Bughin and D. Aharon, "The Internet of Things: Mapping the Value Beyond the Hype," McKinsey Global Institute, 2015.

[3] O. M. Alliance, "Lightweight Machine to Machine Technical Specification, Candidate," 2015.

[4] A. Bröring, J. Echterhoff, S. Jirka, I. Simonis, T. Everding, C. Stasch, S. Liang and R. Lemmens, "New Generation Sensor Web Enablement," *Sensors,* vol. 11, pp. 2652-2699, 2011.

[5] J. Swetina, G. Lu, P. Jacobs, F. Ennesser and J. Song, "Toward a standardized common M2M service layer platform: Introduction to oneM2M," *IEEE Wireless Communications,* vol. 21, pp. 20-26, 2014.

[6] A. Bröring, S. Schmid, C.-K. Schindhelm, A. Khelil, S. Kaebisch, D. Kramer, D. Le Phuoc, J. Mitic, D. Anicic and E. Teniente, "Enabling IoT Ecosystems through Platform Interoperability," *IEEE Software, forthcoming,* 2017.

[7] J. Soldatos, N. Kefalakis, M. Hauswirth, M. Serrano, J.-P. Calbimonte, M. Riahi, K. Aberer, P. P. Jayaraman, A. Zaslavsky, I. P. Žarko and others, "OpenIoT: Open source internet-of-things in the cloud," in *Interoperability and Open-Source Solutions for the Internet of Things*, Springer, 2015, pp. 13-25.

[8] A. Bassi, M. Bauer, M. Fiedler, T. Kramp, R. Van Kranenburg, S. Lange and S. Meissner, Enabling things to talk, Designing IoT solutions with the IoT Architectural Reference Model, Springer, 2013.

[9] F. Ramparany, F. G. Marquez, J. Soriano and T. Elsaleh, "Handling smart environment devices, data and services at the semantic level with the FI-WARE core platform," in *IEEE International Conference on Big Data*, 2014.

[10] J. Soldatos, R. Aikaterini and J. Kaldis, VITAL - Virtualization Architecture and Technical Specifications, vol. D2.3, European Commission - FP7, 2015.

[11] P. Barnaghi, R. Tönjes, J. Höller, M. Hauswirth, A. Sheth and P. Anantharam, "Citypulse: Real-time iot stream processing and large-scale data analytics for smart city applications," in *Europen Semantic Web Conference (ESWC) 2014*, 2014.

[12] A. M. Houyou, H.-P. Huth, C. Kloukinas, H. Trsek and D. Rotondi, "Agile manufacturing: General challenges and an IoT@Work perspective," in *17th IEEE International Conference on Emerging Technologies & Factory Automation (ETFA 2012)*, 2012.

[13] World Wide Web Consortium (W3C), "Web of Things (WoT)," [Online]. Available: https://www.w3.org/WoT/.

[14] J. Mineraud, O. Mazhelis, X. Su and S. Tarkoma, "A gap analysis of Internet-of-Things platforms," *Computer Communications,* 2016.

[15] G. Klyne and J. J. Carrol, *Resource Description Framework (RDF): Concepts and Abstract Syntax, W3C Recommendation,* W3C, 2004.