BIG IoT – Bridging the Interoperability Gap of the Internet of Things

Deliverable 2.4.b:
High-level architecture specification
Version 1.0
Delivery Date: 30.09.2017

This project has received funding from the European Union’s Horizon 2020 research and innovation program under grant agreement No 688038.
# Document Information

<table>
<thead>
<tr>
<th><strong>Responsible Person and Affiliation</strong></th>
<th>Stefan Schmid (Robert Bosch GmbH)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Due Date / Delivery Date</strong></td>
<td>30.09.2017</td>
</tr>
<tr>
<td><strong>State</strong></td>
<td>Final</td>
</tr>
<tr>
<td><strong>Reviewers</strong></td>
<td>Lars Mikkelsen (AAU)</td>
</tr>
<tr>
<td></td>
<td>Juan Hernández Serrano (UPC)</td>
</tr>
<tr>
<td><strong>Version</strong></td>
<td>1.0</td>
</tr>
<tr>
<td><strong>Confidentiality</strong></td>
<td>Public</td>
</tr>
</tbody>
</table>
# List of Authors

<table>
<thead>
<tr>
<th>Organisation</th>
<th>Authors</th>
<th>Main organisations’ contributions</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOSCH</td>
<td>Stefan Schmid</td>
<td>Lead of architecture discussions</td>
</tr>
<tr>
<td></td>
<td>Denis Kramer</td>
<td>First draft of section 2, 3, 4, 6, 7, 8 and 9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Contributions to section 1, 5</td>
</tr>
<tr>
<td>Siemens</td>
<td>Arne Broering</td>
<td>First draft of section 1</td>
</tr>
<tr>
<td></td>
<td>Andreas Ziller</td>
<td>Contributions to section 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Contributions to architecture discussions</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Contributions to architecture discussions</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Contributions to section 7</td>
</tr>
<tr>
<td>ATOS</td>
<td>Martin Lorenz</td>
<td>Contribution to architecture discussions</td>
</tr>
<tr>
<td>CSI</td>
<td>Luca</td>
<td>Contribution to architecture discussions</td>
</tr>
<tr>
<td>TUC</td>
<td>Yong Wang</td>
<td>Contribution to architecture discussions</td>
</tr>
<tr>
<td>NUIG</td>
<td>Achille Zappa</td>
<td>Contribution to architecture discussions</td>
</tr>
</tbody>
</table>
# Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>BCN</td>
<td>Barcelona (pilot)</td>
</tr>
<tr>
<td>BIG IoT</td>
<td>Project title: Bridging the Interoperability Gap of the Internet of Things</td>
</tr>
<tr>
<td>DOA</td>
<td>Description of Action</td>
</tr>
<tr>
<td>IoT</td>
<td>Internet of Things</td>
</tr>
<tr>
<td>LIB</td>
<td>Software Library</td>
</tr>
<tr>
<td>NG</td>
<td>Northern Germany (pilot)</td>
</tr>
<tr>
<td>PIE</td>
<td>Piedmont (pilot)</td>
</tr>
<tr>
<td>SDK</td>
<td>Software Development Kit</td>
</tr>
</tbody>
</table>
# Table of Contents

1. **Introduction** ........................................................................................................... 7
   1.1. Scope and structure of the document ..................................................................... 9
   1.2. Executive Summary ............................................................................................... 9

2. **Definitions and Vocabulary** .................................................................................. 11
   2.1. BIG IoT Domain Model ..................................................................................... 17

3. **Scope** ..................................................................................................................... 19

4. **Use Cases and Requirements** .............................................................................. 22

5. **Methodology** ......................................................................................................... 25

   6.1. Architecture Concepts ....................................................................................... 27
   6.2. Architecture Assumptions and Challenges ......................................................... 34
   6.3. High-level Design Decisions ............................................................................. 39
   6.4. High-level Architecture Description ................................................................... 40

7. **BIG IoT Architecture** ........................................................................................... 43
   7.1. Building Block View ......................................................................................... 43
   7.2. Integration Modes ............................................................................................... 49
   7.3. Interfaces ............................................................................................................ 52
   7.4. Semantic Models and Framework ....................................................................... 57
   7.5. Runtime View ..................................................................................................... 65
   7.6. Deployment View ................................................................................................. 72

8. **Proof-of-Concept Implementation and Demonstrator** ......................................... 75

9. **Conclusions and Outlook** ..................................................................................... 81

References ...................................................................................................................... 83

ANNEX I .......................................................................................................................... 63
ANNEX II

List of Figures

FIG. 1: The BIG IoT Domain Model................................................................. 18
FIG. 3: Clarification of the scope of the BIG IoT Architecture ......................... 20
FIG. 4: Core architecture concepts and terminology, namely Offerings, Offering Providers, Offering Consumers, Queries and Filters ................................................................. 27
FIG. 5: Overview of the target IoT platform types ........................................... 32
FIG. 6: Simplified high-level architecture ...................................................... 41
FIG. 7: Building Block View of the BIG IoT Architecture – Release 1.............. 44
FIG. 8: BIG IoT Semantic Modelling Layers .................................................. 57
FIG. 9: BIG IoT Semantic Model Interplay .................................................... 59
FIG. 10: Architectural details on the Integration of GraphQL and the BIG IoT Semantic Framework .......................................................... 61
FIG. 11: Message sequence diagram for registration of an offering on the marketplace ...... 66
FIG. 12: Message sequence flow of discovery and subscriptions of an offering .......... 68
FIG. 13: Message sequence flow for accessing the resources of an offering ............ 70
FIG. 14: An exemplary deployment scenario of the BIG IoT architecture components in a smart city context ................................................................. 74
FIG. 15: Overall Demonstration Scenarios – including Stakeholders, Demo Data Providers and Applications ................................................................. 75
FIG. 16: BIG IoT Marketplace Web Portal UI for creating new Offerings .......... 76
FIG. 17: Java Source code extract of Demo Parking Space Provider using the Provider Lib ... 77
FIG. 18: Smart City Dashboard with initial Parking Information and Air Quality Data Offerings integrated ................................................................. 79
FIG. 19: Smart City Dashboard after discovery, subscription and integration of a newly created Parking Information Offering .................................................................................................................................................. 80

List of Tables
TAB. 1: Example scenario to illustrate the core architecture concepts and terminology .... 30
TAB. 2: BIG IoT platform provider feedback on required platform types .......................... 33
TAB. 3: BIG IoT platform provider feedback on desired marketplace functionalities .......... 35
TAB. 4: BIG IoT platform provider feedback on the required integration modes .............. 37
TAB. 5: Evaluation of the applicability of integration modes for the different platform types ............................................................................................................................................................ 39
1. Introduction

The idea of an Internet of Things (IoT) is no more a futuristic vision, but indeed an increasing commercial reality that reaches to various application domains ranging from quantified self and smart home applications, over smarter cities and eHealth systems, to Industry 4.0. Dozens of IoT platforms are upcoming. These include cloud solutions, such as Evrythng\(^1\), ThingWorx\(^2\), Xively\(^3\), or Yaler\(^4\), but also on premise solutions such as Bosch’s IoT Suite\(^5\), as well as thematically or geographically focused platforms, such as the Smart Data platform for the Piedmont region\(^6\). However, up to now, these IoT platforms failed to form vibrant IoT ecosystems. This is mainly due to the large number of stakeholders involved in IoT ecosystems, such as IoT platform providers, service and application providers, developers, and users.

Currently, there is no vibrant collaborative IoT ecosystem, since the market entry barriers are high and the potential gain is low for a single stakeholder. Providers of platforms and services require a simple, established way to offer access to their resources. Marketplaces that enable providers to monetize access to the resources offered by their platforms and services are not yet available. Once these marketplaces are established, developers will be able to easily build IoT services and applications and build their products around these. Revenue streams can then be shared across all contributing entities (i.e., IoT platform and service providers). A key task of a marketplace is to provide extended functionalities to enable the advertising, dynamic discovery, automated orchestration, and negotiation of services to facilitate their usage.

However, before such marketplaces can bring their effect, a serious market barrier needs to be tackled: the missing interoperability. The fragmentation of the IoT and the lack of interoperability prevent the emergence of broadly accepted IoT ecosystems [1]. 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., IETF CoAP [3], OASIS MQTT [4], OMA LWM2M [5], OGC SWE [6], or OneM2M [7]) or even proprietary interfaces. As a result, most existing and emerging IoT platforms offer heterogeneous ways for accessing IoT resources (information and functions).

\(^1\) www.evrythng.com
\(^2\) www.thingworx.com
\(^3\) www.xively.com
\(^4\) https://yaler.net
\(^6\) http://www.smartdatanet.it/presentation.html
This causes interoperability problems when overarching, cross-platform, and cross-domain applications are to be built, and eventually prevents the emergence of vibrant IoT ecosystems. Additionally, it leads to barriers for business opportunities, especially for small innovative enterprises, which cannot afford to offer their solution across multiple platforms. They can only provide applications and services for a small number of platforms, e.g., a traffic information application for an IoT platform of a specific city. This lack of interoperability results in lost business opportunities and prevents innovative business ideas.

In order to address these shortcomings in today's IoT landscape, the BIG IoT project developed an architecture as a foundation for building 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. This will allow new services 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 (e.g. the same smart parking application works on top of a smart city platform in Berlin, in Barcelona and in London).

J. Mineraud et al. [14], who have analysed the gaps in today's IoT platforms and solutions, confirm our observations and the approach we are taking. They highlight the fact that "data and device catalogs" and "billing (based on fixed fees, usage or other metrics) of the end-users of the data is generally missing" and suggest that "marketplace functionality shall be provided by future IoT platforms."

Considering that we are still at an early stage of IoT commercialization and the fact that the number of connected things and platforms is expected to increase significantly in the foreseeable future, it is vital for BIG IoT to consider a solution that is able to evolve over time. The ability for applications and services to discover emerging data/information sources or newly offered services and to integrate them at run-time, such that they can leverage "always the best data sources or service available", is therefore one important design goal for the BIG IoT architecture.
1.1. Scope and structure of the document

This document presents the high-level BIG IoT architecture (Release 2) consisting of

(1) a common Web interface, called the BIG IoT API,
(2) semantic descriptions of resources and services, as well as
(3) a marketplace as the core driver of the ecosystem, providing functionalities such as authentication, semantic discovery and accounting.

This deliverable builds on Deliverable D2.4a and extends the architecture by missing features (e.g. reputation management, reporting, offering access control). Release 2 of the architecture also takes into account initial feedback from our pilot projects using our Release 1 implementation.

In Section 1, the document introduces the project goals and challenges that need to be addressed by the BIG IoT architecture. Section 2 defines the terminology that we have established during the process of defining the project and the first months of working together. Section 3 defines the scope of the work and the BIG IoT architecture. Section 4 introduces the main use cases and high-level requirements that guided the architecture discussions. Section 5 explains the methodology and the approach we have followed to develop the first and second release of the high-level architecture. Section 6 introduces the core concepts that we have developed and that now form the basis for the architecture work. We also present the important design decisions that we have taken, and provide a high-level description of the BIG IoT architecture. Section 7 is the heart of the document and presents the Release 2 of the BIG IoT architecture. Here we present a view and description of the key building blocks as well as their interfaces (7.1-7.3). We also introduce high-level message sequence charts for the key interactions of the system (7.4), as well as a deployment view based on an exemplary smart city scenario (7.5). Section 8 presents a demonstrator of the core architecture components, which has been developed based on our Release 1 implementation of the BIG IoT API and Marketplace, as a means for agile architecture definition and to get early feedback from the relevant stakeholders (i.e. developers, creatives and data providers). Finally, Section 9 concludes the document and presents the outlook.

1.2. Executive Summary

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 monetise IoT resources, and thus foster collaboration among the plethora of stakeholders. Having these two crucial pillars of an IoT ecosystem in mind, we developed the BIG IoT high-level architecture as a foundation to establish IoT ecosystems. The architecture fulfils essential requirements that have been assessed among industry and research organisations as part of the BIG IoT project.

This document presents the BIG IoT Architecture Release 2. It builds on Deliverable D2.4.a and extends the architecture by missing features (e.g. reputation management, reporting and offering access control). Our Release 2 of the high-level architecture was refined and extended according to the initial feedback from first trails with test users and our pilot projects, who used our Release 1 implementation to extend their IoT platforms and services, and developed innovative cross-platform and cross-domain IoT applications and services.

The Release 2 of the BIG IoT architecture also introduces (i.) the semantic models that we use to annotate IoT resources in order to achieve interoperability and discoverability, (ii.) the Triple Store we have deployed in our backend to persist the semantic descriptions, as well as (iii.) the reasoning functionality that we use to facilitate discovery of IoT resources via our Marketplace. Moreover, we introduce the concept of service composition through so-called Recipes, which support developers in composing new applications and services based on existing resources and services offered on the BIG IoT marketplace. Finally, we also present our proof-of-concept demonstrator that we have implemented to validate our core ideas and architecture components, and get early feedback on our solution from our target users (i.e. IoT companies and developers).

Keywords: Internet of Things, Architecture, Interoperability, Marketplace
2. Definitions and Vocabulary

To clarify the meaning of terms in the context of this deliverable the following definitions are provided.

**Accounting**

Accounting collects access related data to *Offerings* (e.g. number of access requests, duration of access sessions, and number of data records accessed) and relates it to the respective *Subscription*.

**BIG IoT API**

A set of specifications for

- *Providers* and *Consumers* to interact with the *BIG IoT Marketplace* to authenticate, register, discover and subscribe to *Offerings*; and perform accounting
- *Consumers* to directly access the *Resources* offered by a *Provider*

The *BIG IoT API* defines the supported communication protocols, data formats, semantic descriptions, etc. In order to facilitate *BIG IoT Applications, Services* and *Platforms* to implement and use the *BIG IoT API*, dedicated *Provider* and *Consumer Libs* (SDKs) are provided for various platforms and programming languages, offering also programming interfaces to developers.

**BIG IoT Application (or short Application)**

An application software that uses the *BIG IoT API* to discover *Offerings* on the *BIG IoT Marketplace*, subscribe to *Offerings* and access the offered *Resources*. A *BIG IoT Application* acts merely as an *Offering Consumer*.

**BIG IoT Application / Service / Platform Developer (or short BIG IoT Developer)**

A software developer that implements or integrates a *BIG IoT Service, Application* or *Platform*.

**BIG IoT Application / Service / Platform / Marketplace Provider or Operator**

The organization that operates a *BIG IoT Application, Service, Platform, Marketplace* instance. It is hereby not relevant if a particular instance is hosted on the provider organization’s own infrastructure or a 3rd party infrastructure.
**BIG IoT Core Developer**

A software developer that implements or extends BIG IoT Marketplace and/or BIG IoT Lib components.

**BIG IoT enabled Platform (or short BIG IoT Platform or just Platform)**

An IoT Platform (or Smart Object Platform) that implements and uses the BIG IoT API to register Offerings on the BIG IoT Marketplace and provide access to the offered Resources. A BIG IoT Platform acts merely as an Offering Provider.

**BIG IoT Marketplace**

The BIG IoT Marketplace allows Providers to register their Offerings (based on semantic descriptions) and Consumers to discover relevant Offerings (based on semantic queries) at runtime. It also provides accounting support for Consumers and Providers to track the amount of resources accessed, as well as a web portal for developers and administrators to support the implementation and management of their Applications, Services, and Platforms.

**BIG IoT Organization (or short Organization)**

Participants in the BIG IoT ecosystem are organized in Organizations. Those Organizations are responsible for managing their Providers (with registered Offerings) and Consumers (with registered Queries). Organizations consist of one or more Users to perform that management.

**BIG IoT Recipe (or short Recipe)**

A recipe provides description of the composition of offerings. It is a specification of requirements of an added-value service, and hence it represents a template that can be fulfilled by multiple offerings.

**BIG IoT Semantic Application Domain Model**

The semantic application domain model defines an application-specific vocabulary that builds on the BIG IoT Semantic Core Model, and can be used for annotating Offering Descriptions.

**BIG IoT Semantic Core Model**

The semantic core model specifies all important domain concepts in BIG IoT project including all basic conceptual entities and their relationships. This semantic core model is used as basis
for (1) the Offering Description to define the capabilities of offerings (provided by IoT platforms or services), (2) the Offering Queries (provided by IoT applications and services), and (3) the underlying data model of the BIG IoT Marketplace.

**BIG IoT Semantic Recipe Model**

The semantic recipe model defines a model for BIG IoT Recipes. Such recipes are specifications of requirements of added-value services, and hence they represent templates that can be fulfilled by multiple offerings. All terms and their relations, required for specifying recipes, are defined in semantic recipe model.

**BIG IoT Service (or short Service)**

A BIG IoT Service implements and uses the BIG IoT API to consume and/or provide Offerings via the BIG IoT Marketplace. A BIG IoT Service can act as both, an Offering Consumer and Provider. It typically consumes basic Information or Function in order to offer "higher-value" Information or Functions on the BIG IoT Marketplace.

**BIG IoT User (or short User)**

Part of a BIG IoT Organization that has an account on the BIG IoT Marketplace and can manage the entities of his Organization there.

**Billing**

*Billing* collects *Charging* data and creates invoices.

**Charging**

*Charging* is based on the collected *Accounting* data. The Charging Service multiplies the accounting data with the respective *Price* data of an *Offering* and takes into account special Consumer (group) pricing models, to compute the final amount to be charged.

**Device-level BIG IoT enabled IoT Platform (= Device-level BIG IoT Platform or just Device-level Platform)**

A BIG IoT enabled Platform that is implemented directly on a *Smart Object*, as opposed to on a backend or cloud infrastructure.
**Endpoint**

An *Endpoint* in the context of BIG IoT is a web-based interface for consumers to access Offerings via a Provider. An *Endpoint* description consists of properties like *Endpoint* type and URI.

**End User**

Users of a *BIG IoT Application* are called *End Users*. An *End User* is typically an employee of an Enterprise, SME or Organization (e.g. City Authority), but not limited to that. *End Users* are not part of the BIG IoT ecosystem and are serviced by the *Application Operator* of their Application.

**Function**

Functionality that can be invoked by *Consumers* and is provided by

- a task on an actuator (as part of an *IoT Platform*)
- a *Service* that provides some computational functions or higher level functionality delegating to one or more lower level *Functions*

**Information**

Data provided to *Consumers* by

- a sensor (as part of an *IoT Platform*)
- a *Service* that takes one or more *Information* sources and combines them to provide some added value

**IoT Service (or short Service)**

Software component enabling interaction with resources through a well-defined interface in order to access or manipulate information or to control entities. An *IoT Service* can be orchestrated together with non-IoT services (e.g., enterprise services). Interaction with the service is done via the network. (based on [IoT-A])

**IoT Platform (= Smart Object Platform)**

A computing and communication system that hosts software components enabling interaction with *Smart Objects* in order to access or manipulate information or to control them.
An **IoT Platform** may be implemented on a backend or cloud infrastructure, or directly on a **Smart Object**. Interaction with the platform is done via the network.

**License**

The Provider of an **Offering** can choose the **License** terms for the provided **Information**.

**Offering**

BIG IoT enables **Providers** to offer or trade access to **Information** and **Functions** with **Consumers** via the **Marketplace**. An **Offering** is defined by an **Offering** description, which describes a set of **Resources** offered on the **Marketplace**. It typically encompasses a set of related **Information** or **Functions**. An **Offering** description provides a semantic description of the **Resource(s)** provided to a **Consumer** once the **Offering** is accessed. The description also entails context and meta-information about the **Offering**, including information like the **Region** (e.g. a city or spatial extent) where the **Resource(s)** relate to, the **Price** for accessing the **Resource(s)**, the **License** of the **Information** provided, input & output data fields, etc.

**Offering Consumer (or short Consumer)**

A **BIG IoT Application** or **Service** that is interested to discover and access IoT resources in order to provide a new service or function. A **Consumer** discovers and subscribes to relevant **Offerings** via the **BIG IoT Marketplace**, and accesses the offered resources via the **BIG IoT API**.

**Offering Provider (or short Provider)**

A **BIG IoT Platform** or **Service** that wants to offer or trade IoT resources via the **BIG IoT Marketplace**. A **Provider** registers its **Offering(s)** on the **BIG IoT Marketplace**, and provides access to the offered resources via the **BIG IoT API**.

**Offering Query (or short Query)**

**Consumers** are able to discover **offerings** of interest on the marketplace by providing an (**Offering**) **Query**. A **Query** describes the properties of **Offerings** a client is interested in (**Offering** type, input & output data fields, **Price**, **License**, ...)

**Physical Entity**

Any physical object that is relevant from a user or application perspective. [IoT-A]
**Price**

The *Provider* of an *Offering* can choose the charging model (e.g. *free* or *per month* or *per access*) and amount of money (if applicable) a *Consumer* has to pay when accessing the offered *Resources*.

**Resource**

Abstraction for either *Information* or *Function*.

**Service-level Measurement Reporting (or short Reporting)**

Service-level measurement reporting is performed by the *BIG IoT Consumer Lib* in order to inform the *BIG IoT Marketplace* about measurable service level metrics (e.g. response time, throughput, failure rate) related to *Offerings* and *Providers*. These reports are used by the marketplace to compute a rating of an Offering and Provider, which is provided to *Consumers* as part of the discovery process.

**Smart Object (= Thing)**

A *Device* able to compute and communicate information about itself or related artefacts (*Physical Entities*) to other devices or computer applications; a *Smart Object* is typically attached to or embedded inside a *Physical Entity*. *Smart Objects* either monitor a *Physical Entity* (sensing) or interact with the physical world through actuators (actuation). Those functions can be either controlled autonomously by local computations or triggered from remote.

**Subscription**

Agreement to access the *Resource(s)* of a single *Offering*. This comprises:

- a *Consumer*’s willingness to access the *Offering* (he checked *License*, service level, rating, description, ...)
- the *Consumer*’s consent to pay for the access to the *Resources* (according to the specified *Price*), if applicable
2.1. BIG IoT Domain Model

The following figure illustrates the **BIG IoT Domain Model**. The core entities of the **BIG IoT Architecture** are illustrated in black, namely, the **BIG IoT Marketplace** (in the center) and the **BIG IoT Application, Service** and **Platform**, who act as either a **Provider** or a **Consumer** of IoT resources (i.e. information or functions) or both.

Besides the Marketplace, the **BIG IoT API** (highlighted in blue) is a core feature of the architecture. It constitutes the basis for all interactions and communications within the **BIG IoT Ecosystem**. Regular IoT applications, services and platform become **BIG IoT Applications, Services** or **Platforms**, as they implement and use the **BIG IoT API** in order to register and discover available IoT resources on a **BIG IoT Marketplace**, and access the resources directly. It should be noted that we use the term IoT Platform and Smart Object Platform interchangeably.

The **BIG IoT API** supported interactions are illustrated in blue. Dashed lines are used to indicate **control-plane** interactions with the **BIG IoT Marketplace**, namely to register resource offerings (by **Providers**) and to discover them (by **Consumers**). Solid lines are used for the **data-plane** interactions between **Providers** and **Consumers**, namely for the access to the resources. This separation has been made to indicate the different nature of the communication, i.e., signalling vs. data access, and to consider their diverse architectural requirements.

Finally, the main stakeholder of the **BIG IoT Ecosystem** are illustrated in green, namely **Developers** and **Providers** of **BIG IoT Applications, Services** and **Platforms, Providers** of **BIG IoT Marketplaces** as well as **Users** of an **BIG IoT Application**. The term **BIG IoT Core Developers** is used for any software developers who contributes to the development of the **BIG IoT API and Marketplace**.
Note that the BIG IoT API also includes the direct interfaces between a consumer and provider for accessing resources. This implies that providers (i.e. BIG IoT Services or Platforms) not only use the BIG IoT API for interacting with the Marketplace, but they also implement and provide the access interface part of the API, so that consumers can gain direct access to the resources offered by a provider.
3. Scope

The scope of this document is to define the high-level architecture for BIG IoT Release 1 (Rel-1). In line with the overall project goal: bridging the interoperability gap of the IoT, the scope of this deliverable is to provide an architecture for enabling cross-platform and cross-domain interoperability in order to achieve the goal of lowering market entry barriers for developers.

In particular, the architecture aims to support following five interoperability patterns, which are illustrated in Figure 2.


**Pattern 1 - Cross Platform Access:** The goal of this pattern is to hide that an application or service accesses resources (information or functions) from different platforms through the same interface specification. The challenge of realizing this goal lays in allowing applications or services to discover platforms with relevant information, and enabling platforms that are potentially from different providers to have the same interface and use the same formats to communicate data.

**Pattern 2 - Cross Application Domain Access:** This pattern extends the “Cross Platform Access” pattern. The goal is that services/applications are able to access information and functions not only from different platforms, but also from platforms, which host information from multiple verticals or application domains.
Pattern 3 - Platform Independence: The goal of this pattern is to allow a single application or service to be used on top of different IoT platforms (e.g. in different regions). For example, these can be multiple deployments of a “smart parking” service used in two different geographic regions, which utilize different platforms with information about parking spots.

Pattern 4 - Platform-Scale Independence: The goal of this pattern is to hide different platform scales towards the connecting services and applications. So called server-level platforms are platforms with many devices connected (e.g. a cloud platform), whereas device-level platforms grant direct access to devices (e.g. a mobile device or an edge gateway).

Pattern 5 - Higher-level Service Facades: This pattern extends the interoperability requirements from platforms to higher-level services. The idea is that not only platforms but also services can offer information and functions via a common interface. Thereby, a service acts as a façade towards an IoT platform and accesses the offered information or functions to provide value-added functionalities.

In order to address these five interoperability patterns, the scope of the architecture is on the functionality and interfaces ("the glue") that are required to allow IoT applications, services and platforms to discover each other, to access the provided resources, and to collaborate in order to provide innovative IoT applications and services, while at the same time lower the cost for development and integration for all stakeholders. Figure 3 illustrates this graphically.
In order to avoid any misunderstanding, it is clarified here that the BIG IoT project does not target to develop a new IoT platform architecture or to substitute existing IoT platforms, but merely focuses on "the glue for interoperability" among existing or emerging IoT applications, services and platforms, as outlined above.

At the core of the BIG IoT architecture are therefore the following components:

1. The **BIG IoT Marketplace**, which allows providers of IoT resources to register and advertise their offerings, and consumers to discover them. In order to sustain such a marketplace, it also requires the necessary functionality for authentication and access control of consumers and providers, as well as accounting for the resource exchange enabled through the marketplace.

2. The **BIG IoT API**, which defines the supported communication protocols, data formats, semantic descriptions, etc. for providers and consumers to interact with the marketplace, and for consumers to access the resources offered by a provider.

The target architecture and technologies developed by the project are intended to provide the baseline for an open IoT ecosystem that enables existing IoT platforms, services and applications to interoperate and its providers to collaborate and make business. The ecosystem will be based around the open standard and open source technology that are developed by the project, similar to HTTP/HTML and open source web servers for the World Wide Web.

This document (Deliverable D2.4.b) presents the BIG IoT Release 2 architecture. It builds on Deliverable D2.4.a – Release 1 of the high-level architecture specification. The final release (Release 3) of the high-level architecture will be provided in future versions of this specification, namely in Deliverable D2.4.c.
4. 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.

Note: A complete list of the relevant use case scenarios along with a detailed description is provided in Deliverable D2.2.a: "Use Case Definition" [20]. Likewise, the full set of architectural requirements, that guided the high-level architecture design is captured in Deliverable D2.3.a: "Requirements Analysis and Specifications" [21].

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. Both, the registration as well as the discovery of offerings need to be supported at run-time in order to allow consumers to leverage emerging resources as they become available.

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. Offering composition

It shall be enabled to compose multiple offerings to form higher-level offering. Templates, which we call Recipes, should guide users to create matching compositions. After the Recipe instantiation, an application code shall be generated based on its semantic description. The application code is then runnable on a Cloud-platform, a mobile device or user’s computer. In this way, a Recipe will improve the utilization of IoT resource offerings traded via the marketplace. Moreover, it will promote rapid application development, since a Recipe can be used to create applications easily in an automated fashion – in contrast to creating applications from scratch manually. Furthermore, Recipes shall be discoverable and re-composable with other Recipes, or documented and shared for later application purposes on the marketplace.

6. 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.
5. Methodology

This section describes the methodology that was followed to derive the architecture specification for Release 1 and 2 of BIG IoT.

The scope of the BIG IoT Marketplace was initially not clear and partners in the project had different ideas and understanding what the functionality should include. In a first step, the architecture task therefore evaluated the possible marketplace options, namely a marketplace to offer and exchange IoT resources (information and functions), to trade and download software, and to offer hosting services for applications and services. During a dedicated architectural workshop, the primary role of the marketplace was identified and agreed upon. Based on the partners as well as the overall project priorities, it was decided to focus the first release of the BIG IoT Marketplace on the concept of an “IoT resource exchange”, where provider of IoT resources can publish their offerings and consumer can discover them.

In parallel, the BIG IoT Ecosystem use cases collected for Deliverable 2.2.a [20] have been analysed to derive vital requirements for the architecture. Initially, a set of more than 20 use cases have been considered. However, in order to focus the architecture efforts for the first release, a core set of around 10 use cases were selected based on prioritization through the project partners (see Section 4).

Members of the architecture task force (Task 2.4) met on a regular basis to discuss and evaluate the current state of the BIG IoT architecture. It was proposed to follow an iterative approach for the architecture development, which consists of gathering requirements, incorporating them into the architectural diagram, gathering feedback during the regular meetings and determining new requirements based on the provided feedback. As a starting point of discussion, an initial architecture was proposed that included, on a high level, the elements: IoT platforms, services, applications and the marketplace, as well as the APIs for interacting with the marketplace. Simultaneously, an API for communication between resource consumers and providers was also proposed.

To identify improvements for the first draft and to collect further functional requirements regarding the BIG IoT API and marketplace architecture, a survey among the platform providers in the project was conducted. This survey was done in the project’s internal discussion forum, and raised specific questions such as, what are the key functionalities of the marketplace (see Section 6.2.1) from a platform provider perspective, or which platform integration
modes are required (see Section 6.2.2). Different options were proposed for the platform providers to choose from according to their preferences. The collected feedback was incorporated in the further architectural development.

Furthermore, it was necessary to identify relevant target IoT platform types that the architecture should consider. First, five platform types were identified (see Section 6.1.2). A survey among the partners who implement BIG IoT enabled platforms was conducted to determine which integration modes are realistic for each type. This survey was again realized through the internal discussion forum of the project. A subsequent discussion took place to reflect the survey results and to establish the relevant integration modes for the project. The architecture was then extended to include these modes as well.

The first release of the high-level architecture for BIG IoT was published in M12 (end of December 2016). Following this, we reviewed the achievements and analysed the suitability of the architecture while developing and testing the Release 1 implementation of the BIG IoT API and Marketplace (from M13-M18). Starting from M16, we used our bi-weekly architecture task force meetings to discuss missing functionalities and shortcomings of the Release 1 architecture, and to define the high-level architecture extensions for this Release 2 specification.

In M17 (May 2017), we also organised a first Hackathon around the BIG IoT API and Marketplace at the BuildingIoT Conference 2017 in Heidelberg, Germany. The results and feedback from the developer teams was generally positive, yet we made valuable observations, which we addressed in Release 2. The updated high-level architecture of BIG IoT Release 2 is presented in section 7 of this document.

In order to validate the current architecture proposal, further Hackathons and workshops are currently planned, and a dedicated developer survey is launched. The results of those activities will be taken into account for our third release of the BIG IoT architecture, which will be provided in Deliverables D2.4.c in M27.

This section defines the relevant concepts, architectural assumptions and high-level design decisions that influence and guide the BIG IoT architecture work.

6.1. Architecture Concepts

6.1.1. Offerings, Queries and Filters

This section introduces the core concepts and interaction patterns required for the marketplace to enable registration and discovery of resource offerings. Figure 4 defines the generic concepts that we identified within an IoT ecosystem and the interactions between them. The core concepts are: **Offerings**, **Offering Provider** (in short **Provider**) and **Offering Consumer** (in short **Consumer**), and the interactions of registering and discovering offerings via a marketplace, and accessing the resources offered by a provider. In addition, the concept of **Recipes** is introduced.

![Diagram of core architecture concepts and terminology](image)

**FIG. 4**: Core architecture concepts and terminology, namely Offerings, Offering Providers, Offering Consumers, Queries and Filters

© 2017
An offering encompasses a set of IoT resources, typically a set of related information (e.g. low-level sensor data like temperature or aggregate information across an region) or functions (e.g. actuation tasks like open a gate or computational functions like compute route), that are offered on a marketplace.

Providers register their offerings on a marketplace and provide 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 order to increase interoperability between different IoT platforms, the offering description is provided in a machine interpretable manner, e.g., based on RDF [15] models. In addition, a semantic type or category of the offering, e.g. "Parking" or "Environmental" is provided. Optionally, the offering description may include further sub-categories, such as, for example "Air Temperature" or "Air Pollution". The offering description also includes all relevant communication metadata (e.g., endpoint URI, protocol type and method, etc.) to access the offering by a consumer. As a default vocabulary set, the offering description includes a local identifier (unique to a provider), a name of the offering, the semantic category and optional sub-categories, as well as the input and/or output data. 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 offering access control list, etc.

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 offerings, such as the semantic category or sub-categories, the required input and output data, the maximum price, the accepted licenses, 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. Since the discovery can take
place at run-time, a consumer is able to identify and subscribe to newly offered resources as they emerge. Finally, to limit the data to be requested upon accessing an offering, a consumer can also provide input data as part of the access request, which are then used to filter the resources on the provider end and reduce the number of data records delivered to the consumer.

A **BIG IoT Application** can also be created by instantiating a *Recipe* with offerings on a marketplace that fulfil the Recipe ingredient requirements. A semantic description is created for a Recipe instance with the offerings selected during instantiation. On a marketplace, there can be many offerings from diverse IoT platforms or services that match to Recipe ingredient requirements. In such cases, the user can select an offering based on its spatial constraints or price, recommendations, licensing information etc. Therefore, a Recipe once created can be used multiple times to create applications customised to a user, with matching offerings from diverse IoT platforms and services.

Table 1 presents a concrete example of an offering, a provider, a query, a consumer, and a filter.

<table>
<thead>
<tr>
<th>Term</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Offering</td>
<td>Parking Information in Berlin, including data like the geo location or address of the parking lot, the type of parking lots, available spots, occupied spots, etc.</td>
</tr>
<tr>
<td>Offering Provider (Instance)</td>
<td>VMZ’s IoT Platform (i.e. the run-time instance) that collects parking IoT data around Berlin from sensors, parking lot providers, etc. and processes them to offer higher-level parking information.</td>
</tr>
<tr>
<td>Platform Provider (Organization)</td>
<td>VMZ</td>
</tr>
<tr>
<td>Offering Description</td>
<td>Semantic description (e.g. JSON-LD) for the offered resource, i.e. Parking Information, including data like the price of accessing that information, the license, etc.</td>
</tr>
<tr>
<td>Offering Consumer (Instance)</td>
<td>SEAT’s In-car Smart Parking Application (running on the Head Unit)</td>
</tr>
</tbody>
</table>
### Application Provider (Organization)
- SEAT

<table>
<thead>
<tr>
<th>Offering Query</th>
<th>Semantic Query (e.g. JSON-LD or SPARQL) for information of interest, e.g. <em>Parking Information</em> in the car's current location (e.g. city of Berlin)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Offering Filter</td>
<td>Filter that defines the exact information of interested, e.g. <em>Parking Information</em> in the car's immediate surrounding (e.g. Kurfürstendamm or geo location)</td>
</tr>
</tbody>
</table>

**TAB. 1:** Example scenario to illustrate the core architecture concepts and terminology

## 6.1.2. IoT Platform Types

A central goal of the BIG IoT architecture is to facilitate the integration of IoT platforms through the BIG IoT API into the BIG IoT ecosystem. Both infrastructure as well as device-level platforms are targeted.

From an architectural perspective, specifically considering the implementation of the BIG IoT API and integration with marketplaces, we have identified the following five types of IoT platforms:

**Type 1:** Server Infrastructure or Cloud based IoT Platform
- Type 1 platforms are assumed to be "always online" and anytime accessible by applications or services via the Internet.

**Type 2:** Device-level IoT Platform, hosted on devices that are **unconstrained** with respect to communication, compute and memory resources
- Type 2 devices are also assumed to be "always online" and anytime accessible by applications or services via the Internet, whereby connectivity and communication resources is assumed to be charged based on a "flat-rate" plan (like for type 1).

---

7 **Unconstrained** in this context means that the device will be able implement/use the BIG IoT API, which will be based on typical Web/Internet technologies (e.g. HTTP, WebSockets). An example of an unconstrained device is a Raspberry Pi. Also other devices that are able to run Linux and support typical Web/Internet technologies are considered unconstrained in this context. A micro-controller based Sensor is not considered unconstrained.
Type 3: Device-level IoT Platform, hosted on devices that are unconstrained with respect to communication, compute and memory resources, but are "not always online"

- Type 3 devices cannot be assumed to be "always online", as they might be energy constrained (and thus in energy-saving mode) or mobile (and thus at times out of coverage).

Type 4: Device-level IoT Platform, hosted on devices that are unconstrained with respect to communication, compute and memory resources, but are connected to the Internet via a "pay-per-use" plan

- Type 4 devices are charged based on a "pay-per-use" plan (i.e. not flat rate). It is thus desirable to minimize the network traffic towards/from such a device.
- Type 4 devices are often also of Type 3.

Type 5: Device-level IoT Platform, hosted on devices that are constrained with respect to communication, compute and/or memory resources

- Type 5 devices are typically low-cost sensors, using a micro-controller, and thus constrained with respect to the implementation of the BIG IoT API, as common Web API frameworks cannot simply be adopted. Additional efforts are expected to implement the BIG IoT API on such devices.
- Type 5 devices are often at the same time "not always online" due to energy constraints (i.e. type 3) or might be connected via a "pay-per-use" plan (i.e. type 4).

Figure 5 illustrates the five different IoT platform types.

---

*Constrained* in this context means with respect to the implementation of the BIG IoT API, which will be based on typical Web/Internet technologies (e.g. HTTP, WebSockets). An example of constrained devices are low-cost sensors, using a micro-controller. A Raspberry Pi is not considered a constrained device.
Platform types 1 and 2, although quite different in scale (infrastructure level vs. device level), have in common that they are both "always online" and have sufficient compute and memory resources to implement and offer Web based APIs for the interaction with the marketplace and other BIG IoT applications or services (as consumers).

Platform types 3 and 4 also have sufficient compute and memory resources to implement and offer Web based APIs for the BIG IoT related interactions, but they both are limited with respect to the connectivity and communication resources respectively. While platform type 3 lacks availability with respect to connectivity, type 4 is limited with respect to the resource consumption on the communication link. Both platform types would thus benefit from an infrastructure based proxy node, which is always available and can help reduce the communication resources on the platform’s backhaul connection.
Platform type 5 differs from the others with respect to the available compute and memory resources. As such, it is assumed that special implementations of the BIG IoT API will be required to connect such constraint and low-cost devices into the BIG IoT ecosystem.

In the context of the BIG IoT project itself, we have the following nine IoT platform represented. Depending on the deployment scenario, they can be classified into the described platform types as follows:

<table>
<thead>
<tr>
<th>Platform Type / BIG IoT Platforms</th>
<th>Bosch Smart City Platform</th>
<th>Bosch BEZIRK Platform</th>
<th>CSI Platform</th>
<th>OpenIoT Platform</th>
<th>World-Sensing Platform</th>
<th>VMZ Platform</th>
<th>Siemens APM Platform</th>
<th>Econais Wubby Platform</th>
<th>Vodafone Platform</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type 1</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td>Type 2</td>
<td>☑</td>
<td></td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td>Type 3</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td></td>
<td></td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td>Type 4</td>
<td>☑</td>
<td></td>
<td>☑</td>
<td></td>
<td></td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td>Type 5</td>
<td>☑</td>
<td></td>
<td>☑</td>
<td></td>
<td></td>
<td>☑</td>
<td>☑</td>
<td></td>
<td>☑</td>
</tr>
</tbody>
</table>

Tab. 2: BIG IoT platform provider feedback on required platform types
6.2. Architecture Assumptions and Challenges

6.2.1. Marketplace Assumptions

In order to check if the high-level goal of the project and the requirements with respect to the BIG IoT Marketplace match the needs of the IoT platform providers represented in the project, we collected the following feedback from the 9 IoT platform providers involved in the project. There are 7 cloud- or infrastructure-level platforms: Bosch’s Smart City platform, based on the Bosch IoT Suite, CSI’s Smart Data platform, OpenIoT, VMZ’s TIC platform, Siemens APM platform, and WorldSensing. Further, there are 2 device-level platforms: Bosch’s BEZIRK platform and Econais’ Wubby platform. Specifically, we asked each platform provider to indicate which of the core marketplace functionalities is expected/required for the BIG IoT Marketplace to be an attractive proposition for the respective provider.

The results of the internal platform provider survey are illustrated in Table 3. The results confirm the strong interest from all IoT platform providers to use a marketplace that allows the provider to register and offer their resources via the marketplace, and enable potential consumers (i.e. applications or services) to discover the resources and access them. Seven of the nine platform providers also indicated the need for Identity Management (IdM) functionality provided by the marketplace in order authenticate and authorize the consumers once they access resources on a platform. Note that without that functionality, the platform must provide its own IdM to authenticate and authorize consumers. Without the IdM functionality provided through the marketplace, any consumer of a platform must first sign up directly with the platform provider and establish an account before access to the resources discovered via the marketplace can be granted.

Five platform providers also indicated interest in accounting support by the marketplace, meaning that the marketplace will collect accounting records from the platform and/or application or service that accesses resources on the platform.
With respect to charging and billing functionality, only 4 platform providers showed clear interest, with 2-3 being still undecided.

<table>
<thead>
<tr>
<th>Desired Marketplace Functionality / Platform Provider</th>
<th>Bosch Smart City Platform</th>
<th>Bosch BEZIRK Platform</th>
<th>CSI Platform</th>
<th>OpenIoT Platform</th>
<th>World-Sensing Platform</th>
<th>VMZ Platform</th>
<th>Siemens APM Platform</th>
<th>Econais Wubby Platform</th>
<th>Vodafone Platform</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identity Management - for Marketplace customers, as basis for authentication + access control</td>
<td>✔️</td>
<td>✔️</td>
<td>✗️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✗️ but possibly in future</td>
</tr>
<tr>
<td>Exchange of resource offerings - for registration (providers) and discovery (consumers)</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Accounting of consumed resources - as basis for charging</td>
<td>✔️</td>
<td>✔️</td>
<td>✗️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>?</td>
<td>✔️</td>
<td>✗️ but possibly in future</td>
</tr>
<tr>
<td>Charging – for providers/consumers to see their earnings/costs</td>
<td>?</td>
<td>✔️</td>
<td>✗️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>?</td>
<td>?</td>
<td>✗️</td>
</tr>
<tr>
<td>Billing - for providers to invoice their consumers</td>
<td>?</td>
<td>✔️</td>
<td>✗️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>?</td>
<td>?</td>
<td>✗️</td>
</tr>
</tbody>
</table>

Legend: ✔️ Desired, ✗️ Undecided, ✗️ Not needed

The results of this feedback are also used to prioritize the BIG IoT Marketplace functionalities for Release 1 and 2. In particular, it was decided to limit the scope on the following functions, in order of priority:

1. Registration and discovery of resources offerings
2. Identity Management
3. Accounting

The marketplace capabilities for charging and billing are left for Release 3 of BIG IoT.
6.2.2. Platform Integration Modes

For the integration of heterogeneous IoT platforms into IoT ecosystems, we have analysed the needs and constraints of the 9 platforms involved by the BIG IoT project partners. 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 device-level platform providers (i.e. for Type 3 and 4 platforms) require infrastructure-level support to overcome the availability and cost limitations of such platforms.

In order to address challenge 1, we concluded to offer platform (and to the same extend also application and service) developers a BIG IoT API Library or SDK (short BIG IoT Lib) for the common IT platforms and programming languages. This way, a developer will be able to programmatically extend an IoT platform and use the BIG IoT API based on a simple to use programming interface.

To cope with challenge 2, we suggest that affected platform providers develop a gateway service, which bridges between the BIG IoT world on the one hand and the existing platform on the other hand. Such a gateway service would then sit "in between" the marketplace or consumer applications/services, and the existing platform. The BIG IoT Lib would again facilitate the implementation of the BIG IoT API for the gateway service, so that developers have to mainly deal with the proprietary interface towards their own platform. We also envision that open source gateway services will become available for common IoT platform types, allowing developers to simply adopt and configure them for their specific setup. Moreover, for platform providers that do not mind limiting the access to resources on their platform to its current customers and that offer already standard REST-based interfaces to their resources, there is also the possibility to substitute the gateway service with a simpler management service that deals just with the registration of the platform's offering on the marketplace. In that case, access to the resources would be directly terminated by the existing IoT platform. The limitation of course is that consumers must have already an established business relationship with the provider and the necessary access credentials must be available at the consumer end.
In order to deal with challenge 3, we support affected platform providers with a standard proxy service solution and an extended BIG IoT Lib that allows easy integration of their device-level platform with the marketplace and the proxy service. The main functionality of the proxy service will be to store informational resources that have been provided by the device-level platform, and offer them to interested consumers. With respect to tasks or actions that need to be executed on the device level platform, the proxy service will receive and queue them until the devices connect and pull the latest tasks. The response of a task or action will also be proxied by this service. This solution copes with the shortcomings of type 3 and 4 platforms. We envision that the BIG IoT project will provide common, open source proxy service implementations, ready to be deployed by the respective device-level platform providers.

We validated the different integration options with all the platform providers involved in the project. Table 4 shows the feedback of the nine providers with respect to the feasibility of adopting the four different integration modes. The results show that 5 out of 9 platform providers are interested in the API library to extend their platform programmatically, and 2 providers are undecided. In addition, 6 out of 9 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.

<table>
<thead>
<tr>
<th>Platform Integration Mode / Feasibility</th>
<th>Bosch Smart City Platform</th>
<th>Bosch BEZIRK Platform</th>
<th>CSI Platform</th>
<th>OpenIoT Platform</th>
<th>World-Sensing Platform</th>
<th>VMZ Platform</th>
<th>Siemens APM Platform</th>
<th>Econais Wubby Platform</th>
<th>Vodafone Platform</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mode 1: Programmatic Integration (based on the BIG IoT Lib)</td>
<td>✔️</td>
<td>✔️</td>
<td>❌</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>❓</td>
<td>❓</td>
<td>❌</td>
</tr>
<tr>
<td>Mode 2: Gateway Service based Integration</td>
<td>✔️</td>
<td>❌</td>
<td>❌</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>❌</td>
<td>✔️</td>
</tr>
<tr>
<td>Mode 3: Management Service based Integration</td>
<td>❌</td>
<td>❌</td>
<td>✔️</td>
<td>❓</td>
<td>❓</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
</tr>
<tr>
<td>Mode 4: Proxy Service based Integration</td>
<td>✔️</td>
<td>✔️ (for devices not always online)</td>
<td>❌</td>
<td>❓</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>✔️</td>
<td>❓</td>
</tr>
</tbody>
</table>

Legend: ✔️ Feasible, ❓ Undecided, ❌ Not feasible

TAB. 4: BIG IoT platform provider feedback on the required integration modes
Furthermore, we have also evaluated the different integration modes with respect to their applicability for integrating IoT platforms according to the five types identified in Section 6.1.2.

<table>
<thead>
<tr>
<th>IoT Platform Types / Integration Modes</th>
<th>Mode 1: Programmatic (based on the BIG IoT Lib)</th>
<th>Mode 2: Gateway-Service based</th>
<th>Mode 3: Management-Service based</th>
<th>Mode 4: Proxy-Service based</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type 1: Infrastructure-level Platform</strong></td>
<td>✅&lt;br&gt;For platforms that can be programmatically extended</td>
<td>✅&lt;br&gt;For platforms that cannot be programmatically extended</td>
<td>✅&lt;br&gt;For platforms that cannot be programmatically extended</td>
<td>Possible, but no added value</td>
</tr>
<tr>
<td><strong>Type 2: (Unconstrained) Device-level Platform</strong></td>
<td>✅&lt;br&gt;For platforms that can be programmatically extended</td>
<td>Possible if device-level platform cannot be extended, but overhead of running addition GW Service unclear</td>
<td>Possible if device-level platform cannot be extended, but overhead of running additional Mgmt Service unclear</td>
<td>Infrastructure based Proxy Service may improve availability</td>
</tr>
<tr>
<td><strong>Type 3: &quot;Not-always-accessible&quot; Device-level Platform</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Type 4: &quot;Pay-per-use-connected&quot; Device-level Platform</strong></td>
<td>Possible, but may introduce high communication costs</td>
<td>Possible, but may introduce high communication costs</td>
<td>Possible, but may introduce high communication costs</td>
<td></td>
</tr>
<tr>
<td><strong>Type 5: &quot;Memory-and-compute-constrained&quot; Device-level Platform</strong></td>
<td>✅&lt;br&gt;For platforms that can be programmatically extended - low resource Provider Lib is needed</td>
<td>Possible if device-level platform cannot be extended, but overhead of running addition</td>
<td>Possible if device-level platform cannot be extended, but overhead of running addition</td>
<td>Infrastructure based Proxy Service may improve availability</td>
</tr>
</tbody>
</table>
6.3. 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 the registration + discovery of IoT resource offerings.**

The functional scope of a marketplace in an IoT ecosystem can be broad. The scope of the BIG IoT Marketplace was initially not clear, and partners in the project had different ideas and understanding of what the functionality should include. In a first step, the architecture task therefore evaluated the following possible functional options:

- Resource exchange – for IoT resource providers and consumers to register and discover resource offerings and facilitate the resource access;
- 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 urgent and innovative. Based on the partners as well as the overall project priorities, we decided to focus the BIG IoT Marketplace on this.

2. **Consumers access IoT resources directly on the provider infrastructure.**

For scalability reasons and to keep control to IoT resources under full control of the providers, we decided not to store IoT data on a marketplace, but to enable access to resources directly
on the provider end. This design decision has the advantage that the marketplace only requires the ability to publish and discover 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. In other words, the marketplace is only involved in the control plane interactions and the actual user plane access to the resources is handled bi-laterally between the consumer and provider, avoiding any bottlenecks on the marketplace end.

3. Providers and consumers can participate on multiple marketplaces.

In order to avoid a marketplace lock-in\textsuperscript{16}, we decided 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 BIG IoT 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.

6.4. High-level Architecture Description

This section introduces the high-level architecture of BIG IoT based on a simple architecture representation (see Figure 6) in order to describe the core building blocks and interactions.

In the center, we see the BIG IoT Marketplace, as meeting point for ecosystem stakeholders: Organizations (e.g. their developers or administrators) of IoT resource providers (platforms or services) and consumers (services or applications). The Marketplace connects the involved software instances that either offer resources on the marketplace (BIG IoT Providers) or consume resources (BIG IoT Consumers).

As mentioned before, one of the project objectives is to lower the hurdle for developers/organizations to join a BIG IoT ecosystem. As a consequence, BIG IoT provides easy to use Software Development Kits (SDKs) for BIG IoT Consumers and Providers, allowing developers with a short introduction and low implementation efforts to connect their IoT platforms, services and applications to the Marketplace. The project aims to provide ready-to-use- BIG IoT Consumer and Provider Libs for major programming languages (e.g. Java, JavaScript, ...) and relevant platforms (e.g. Cloud servers and embedded platforms).

\textsuperscript{16} By binding a provider or consumer to a single marketplace provider.
The BIG IoT Provider and Consumer Libs offer a simple Application Programming Interface (API) for easy integration with the service or application logic of BIG IoT Providers and Consumers. The BIG IoT programming interface is introduced by the BIG IoT Development Guide [22]. It supports basic functionality for the Consumer and Provider instances to authenticate on the marketplace, to register offerings, to discover offerings and finally to subscribe to offering of interest. Once subscribed, a Consumer is able to access offerings via the Consumer Lib.

High-level Architecture – simplified

![High-level Architecture](image)

**FIG. 6:** Simplified high-level architecture.

Depending on the provider platform, the Provider Lib may also bring up an Endpoint during offering registration, enabling the Provider to offer the data directly via the Provider Lib functionality to authorized Consumers. Note that the Consumer Lib only provides access to an offering once the offering has been subscribed to on the Marketplace. As a result of a successful subscription, the Consumer Lib obtains a secure *Offering Access Token*, which is used in the access request on the Provider. Upon an access request, the Provider Lib first checks if the request includes a valid Offering Access Token (signed by the Marketplace for the specific Consumer that subscribed to the offering). If the Offering Access Token is valid, the Provider Lib will service the request and call the Access Callback function defined by the Provider to fetch the offered IoT resources and deliver them to the Consumer. While a Consumer accesses an offering on the Provider end, both the Consumer and Provider Lib will also send accounting related data to the marketplace, which will be used as the basis for charging and billing, and
to visualize statistics information to the developers/administrators of Providers and Consumers.

Not shown in the figure is the reporting of service level data (e.g. response times, throughput, and failure rate) that are measured and collected by a Consumer when accessing an offering on a Provider end. This information is regularly reported to the marketplace, which in turn stores this information with the Provider and/or Offering Descriptions.

The detailed high-level architecture is presented in Section 7 of this document.
7. BIG IoT Architecture

This section presents the BIG IoT Architecture – Release 2. The architecture is introduced based a building block diagram in Section 7.1. Section 7.2 introduces the currently anticipated platform integration modes. The interfaces among the BIG IoT components are described in Section 7.3. In Section 7.4, we present the semantic models and frameworks used to enable semantic discovery of IoT resources on the BIG IoT Marketplace. Section 7.5 provides the runtime view (message sequence charts) for the main interactions and operations. Finally, Section 7.6 concludes the architecture description with an exemplary deployment view of the BIG IoT architecture in the context of a fictive smart city scenario.

It should be noted that this section repeats many of the core concepts and descriptions provided in earlier section intentionally in order to provide the reader a self-contained description of the BIG IoT architecture.

7.1. Building Block View

This section presents the building block view of the BIG IoT architecture. Figure 7 shows the core building blocks and the interfaces among them.

Note: For better readability, a bigger illustration of the building block view of the high-level architecture is also provided in Annex 1 of this deliverable.

7.1.1. BIG IoT Application

A BIG IoT Application is an Application Software that implements and uses the BIG IoT API (as Consumer) to discover resource offerings (information or functions) on a BIG IoT Marketplace, and to access the resources provided by a BIG IoT Service or Platform (as Providers). The developer of a BIG IoT Application uses the Programming Interface P2 of the BIG IoT Consumer Lib to use the BIG IoT API.

7.1.2. BIG IoT enabled Platform

A BIG IoT enabled Platform (short BIG IoT Platform) is an IoT Platform that implements and uses the BIG IoT API (as Provider) to register offerings on a BIG IoT Marketplace, and to allow BIG IoT Services or Applications (as Consumers) to gain access the offered information or function.
Four different Integration Modes for integrating existing IoT platforms into the BIG IoT Ecosystem have been identified (see Section 7.2 for further details):

**Integration Mode 1:** the platform provider uses the **Programming Interface P1**, offered by the **BIG IoT Provider Lib** to extend an existing or new IoT platform in order to join the **BIG IoT Ecosystem** and allow offering of resources via the **BIG IoT Marketplace**.

**Integration Mode 2:** the platform provider develops and operates a **BIG IoT Gateway Service**, which handles all BIG IoT related interactions with the platform and translates the relevant BIG IoT requests into calls supported by the existing platform. For this, the **BIG IoT Gateway Service** uses the **Integration Interface I1**.
Integration Mode 3: the platform provider develops and operates a BIG IoT Management Service, which handles all BIG IoT Marketplace related interactions. It also translates the relevant BIG IoT requests into calls supported by the existing platform, by using the Integration Interface I2. Access to the resource offerings, however, will be directly provided by the existing platform, based on the already available interface.

Integration Mode 4: the platform provider develops and operates a BIG IoT Proxy Service, which offers the Access Interface A1 to the offered resources. The BIG IoT Proxy Service acts as an "always-available" proxy on behalf of a typically constrained BIG IoT Platform. The proxy stores and offers information that are provided by the constrained platform and is able to queue access requests (e.g. tasks) to functions offered by the constrained platform until the platforms wakes up or reconnects. The registration of offerings on the BIG IoT Marketplace is typically directly initiated by the device-level platform.

7.1.3. BIG IoT Service

A BIG IoT Service is an IoT Service that implements and uses the BIG IoT API to register offerings on a BIG IoT Marketplace (as Provider) and/or to discover and access offerings provided on a BIG IoT Marketplace (as Consumer).

The developer of a BIG IoT Service uses the Programming Interface P1 of the BIG IoT Provider Lib to register offerings on a BIG IoT Marketplace and provide access to its resources for other BIG IoT Services and Applications (as Consumers). The developer of the BIG IoT Service uses the Programming Interface P2 of the BIG IoT Consumer Lib to discover offered resources on a BIG IoT Marketplace and gain direct access to offerings provided by other BIG IoT Services or Platforms (as Providers).

7.1.4. BIG IoT Library

The BIG IoT Library (or SDK) offers a programming interface for developers of a BIG IoT Platform, Service or Application. It consists of a Provider and Consumer part, and will be available for different target platforms and programming languages. It will translate function calls from the respective application or service logic, or the platform code into interactions with the Marketplace, or peer-services and -platforms.
The BIG IoT Provider Lib will allow a platform or service instance (as Provider) to authenticate itself on the Marketplace and register offerings to be offered on the Marketplace via the respective Marketplace Interfaces M1 and M2. The Provider Lib may also (depending on the integration mode) instantiate the Access Interface A1 via which the platform or service will grant Consumers access to the offerings.

The BIG IoT Consumer Lib will allow an application or service instance (as Consumer) to authenticate itself on the Marketplace, to discover available offerings (based on a semantic query), and to subscribe to offerings of interest through the respective Marketplace Interfaces M1, M3 and M4. In case a Consumer subscribes to a desired offering, the Marketplace will provide the necessary credentials for the Consumer Lib to access the subscribed offering on the Provider platform or service. The Consumer Lib will then directly access resources offered by a Provider platform or service through the Access Interface A1. The information or data obtained by the Consumer Lib upon an access request initiated by the application or service instance will be provided to the application or service via the P1 Interface. It should be noted that an access request can be either one-time (i.e. request-response) or continuous (streaming). In the latter case, the Provider will feed further information as they become available to the Consumer until the application or service terminates the continuous access session.

Once a Consumer accesses an offering of a Provider via the A1 Interface, both the Consumer Lib and the Provider Lib will perform accounting on the resource consumption, and send this information via the Accounting Interface M5 to the BIG IoT Charging Service.

7.1.5. BIG IoT Marketplace

The BIG IoT Marketplace consists of the following building blocks:

BIG IoT Marketplace API

All communications and interactions with the Marketplace are provided via the Marketplace API. This covers both the standard Web API for accessing the Marketplace Web Portal, as well as the BIG IoT API specific interfaces for authentication of Providers and Consumers (M1), offering registration (M2), offering discovery (M3), offering subscription (M4), as well as accounting when accessing offerings (M5). The details of these interfaces will be provided in Section 7.3.
BIG IoT IdM

The **BIG IoT Identity Management (IdM)** is the Marketplace component that is responsible to authenticate and authorize **Providers** (BIG IoT Platforms and Services) and **Consumers** (BIG IoT Applications and Services) in the BIG IoT Ecosystem via the *Marketplace Interface M1*.

Upon authentication, the IdM will provide the **Consumer or Provider Lib** (as client) the credentials that are used in any further communications with the Marketplace.

It is assumed that **BIG IoT Developers** as well as **BIG IoT User and Application, Service or Platform Providers** will first register on the **BIG IoT Marketplace** via the *Web Portal* in order to obtain the required information for the authentication (e.g. an API key and/or a certificate).

BIG IoT eXchange

The **BIG IoT eXchange** is the Marketplace component that allows **Providers** to register their offerings on the Marketplace (by providing a semantic description of the offering) via the *Marketplace Interface M2*.

Correspondingly, the BIG IoT eXchange lets **Consumers** discover offerings (by providing a semantic search query of the desired resources) and subscribe to offerings via the *Marketplace Interfaces M3 and M4 respectively*. As such, the **BIG IoT eXchange** provides the necessary functionality to match semantic descriptions of offerings with semantic queries for offerings.

The use of semantic technologies enables the BIG IoT eXchange to perform described semantic matching even in case the 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 by the Marketplace.

BIG IoT Web Portal

The BIG IoT Web Portal allows users of the Marketplace (typically organizations) to register their organizations and create user accounts for their relevant developers and administration personnel to manage the organizations’ **Providers, Consumers**, offerings, queries and subscriptions, as well as charging related information.

Via the Web Portal, developers can download the **BIG IoT Library** (or SDK) for their target platform and programming language, and get access to the relevant documentation of how to use the **BIG IoT Library** and **Marketplace** functionality. The Web Portal also supports developers or administrators, after they have authenticated themselves, to register new
Provider or Consumer instances, define new offerings and queries (for the supported application domains), query and subscribe to relevant offerings, and manage those conveniently via a Web browser.

The Web Portal also allows administrators of an organization to view and analyse the organization's earnings and expenditures with respect to the offerings and subscriptions of its Providers and Consumers respectively.

**BIG IoT Accounting Service**

The BIG IoT Accounting Service obtains accounting information provided by the Consumer and Provider Libs over the M5 interface. The Libs count the number of access requests per offerings, or the number of updates and the duration in case of streaming-based access, as well as the number of output data records and the data volume (payload bytes) delivered to a Consumer. Both the Consumer and Provider Libs send the accounting information based on a "windowed approach" (i.e. after a defined time interval, when a certain number of messages, and/or when a certain price value has been reached). The accounting service checks if the accounting records sent by the Provider and Consumer Libs match. In case of a significant discrepancy, the accounting service informs the Consumer and Provider, and terminates the Offering Subscription of the Consumer.

The accounting service also persists this information in order to compute the Providers' earnings and Consumers' expenses based on the defined pricing model defined in the offering descriptions. This accounting information will be accessible to the responsible people of a Provider or Consumer Organization on the basis of individual Providers, Consumers, and offerings and subscriptions via the BIG IoT Web Portal.

**Triple Store**

All the semantic models such as the core model around the Offering Description, the Offerings itself, the application domain specific terms as well as the Recipe model and Recipes instances are stored in the Triple Store, which runs in the back-end of the marketplace. The BIG IoT eXchange translates incoming queries from the Marketplace API to SPARQL queries in order to answer requests, e.g., to discover matching offerings for an Offering Query.

**Recipe Cooker**

The Recipe Cooker is a graphical user interface to support a developer to create a Recipe, instantiate it, and generate application code from the instantiated Recipe. The tool provides an easy-to-use interface to users for creating customized compositions of IoT offerings. It can
be used as a front-end to a Web-based IoT marketplace with a triple store running in the back-end. The Recipe Cooker tool interacts with the Triple Store in the back-end to discover the semantic data, to download them and to upload semantic descriptions of a Recipe or Recipe instance to it.

7.2. Integration Modes

In order to facilitate IoT Platform providers to join the BIG IoT Ecosystem, the architecture considers various integration modes. Rel-1 of the BIG IoT architecture supports the following four integration modes:

7.2.1. Mode 1: Programmatic integration based on BIG IoT Provider Lib

In Integration Mode 1, the developer of the BIG IoT Platform extends an existing or new IoT platform by means of the Programming Interface P1 of the BIG IoT Provider Lib. As such, in this integration mode, the legacy platform becomes a full-fledged BIG IoT Platform, without the needed to run and operate any other components or service.

The BIG IoT Provider Lib hides all the complexity of interacting with the BIG IoT Marketplace and provides the Access Interface A1 to access the offered resources. Depending on the chosen IoT platform framework and programming language, an adequate version of Provider Lib has to be chosen. The supported access paradigms (e.g. request/response, streaming) and protocols (e.g. HTTP, WebSocket, MQTT) depend on the chosen Provider Lib implementation.

7.2.2. Mode 2: Integration based on BIG IoT Gateway Service

In Integration Mode 2, the provider integrates an existing IoT platform into the BIG IoT Ecosystem by means of a BIG IoT Gateway Service. This integration mode has the advantage that the provider does not need to alter an already available platform (e.g. a purchases off-the-shelf platform). However, the provider has to develop and operate a BIG IoT Gateway Service, which is able to handle any BIG IoT related communication and translates the relevant BIG IoT requests into calls supported by the existing platform. For this, the BIG IoT Gateway Service has to implement the Integration Interface I1. The I1 interface is the original interface of the IoT platform. From a BIG IoT Marketplace or Consumer perspective, the BIG IoT Gateway Service looks like a regular BIG IoT Service or Platform (Provider).
To implement the **BIG IoT Gateway Service**, the developer can use the **BIG IoT Provider Lib** to implement and use the **BIG IoT API**. As such, the developer has to implement merely the Gateway Service Logic that translates any BIG IoT related interactions to the corresponding call to the existing IoT platform.

TBD: The possibility to provide 'ready to use' gateway services that need merely configuration to adapt it to an IoT platform; such gateway services could be provided for major IoT platforms.

### 7.2.3. Mode 3: Integration based on BIG IoT Management Service

In **Integration Mode 3**, the provider integrates an existing IoT platform into the BIG IoT Ecosystem by means of a BIG IoT Management Service. Therefore, neither the existing, legacy interface of the platform has to be changed nor a new interface has to be added. This integration mode, similar to Mode 2, has the advantage that the provider does not need to alter an already available platform. Moreover, it is not required to operate a **BIG IoT Gateway Service** that is on the critical access path between a consumer and the actual IoT platform, as the access to the platform resources is provided directly. The provider is only required to develop and run a **BIG IoT Management Service**, which is able to handle the interactions with the **BIG IoT Marketplace** (i.e., authentication and registration of offerings) on behalf of the existing platform.

While this integration mode looks at a first glance more attractive than Mode 2, it has the following limitations:

1. As any A1 access requests is directly handled by the legacy platform in this mode, the mode is only applicable for the integration of platforms that offer already an access interface that is compatible with the A1 interface (implemented by the **BIG IoT Consumer Lib**).
2. Only Consumers, which have already an account with the IoT platform provider, can gain access to the offered resources, as only the legacy interface is provided by the platform to be integrated, and as such has to control the access based on the existing keys or credentials. Because of this limitation, a Consumer must be aware of this integration mode in order to use locally available keys or credentials to access such a platform. Alternatively, if the legacy platform can be extended or configured to check the offering access token signed by the Marketplace, no exception are needed in the Consumer Lib for access control purposes on such a platform.
3. Accounting for accessing those resources cannot be handled by the **BIG IoT Provider Lib** and Marketplace, as in the other integration modes.

4. The platform can only provide resources offered by the legacy platform "as-is" to **Consumer** applications or services, as there is no way to reformat the data or augment the data by adding semantic labels.

To implement the **BIG IoT Management Service**, the developer can use the **BIG IoT Provider Lib** to handle the interactions with the Marketplace. As such, the developer has to implement merely the **Management Service Logic**, which handles the authentication of the **Provider** platform as well as the registration of the offerings.

### 7.2.4. Mode 4: Integration based on BIG IoT Proxy Service

**Integration Mode 4** has been designed for constrained device-level IoT platforms (see IoT Platform Types 3/4/5) that are not always reachable or want to limit the traffic to/from the device. In that mode, the provider integrates the device-level IoT platforms into the **BIG IoT Ecosystem** by means of a **BIG IoT Proxy Service**. The proxy service acts as an "always-available" proxy on behalf of the constrained devices. The proxy stores and offers information that are provided by the constrained platform and is able to queue access requests (e.g. tasks) to functions until such platforms wake up or reconnect.

For easy extension of such constrained device-level IoT platforms, a special **BIG IoT Lib** for the proxy service integration is provided. It allows the developer to easily interact with the **BIG IoT Marketplace** and the **BIG IoT Proxy Service** via the **Programming Interface P3**. In addition to the normal **P1 interface** functions for **Provider** authentication, offering registration, and accounting, the **P3 interface** will also allow the platform to push information elements to be stored and served via the proxy service over the **PR interface** and to fetch tasks that have been queued by the proxy service on behalf of the constrained platform.

From a **BIG IoT Marketplace** or **Consumer** perspective, the **BIG IoT Proxy Service** looks like a regular **BIG IoT Service or Platform (Provider)**. As such, any limitations of a constrained IoT platform can be overcome through this integration mode. The **BIG IoT Proxy Service** will represent the constrained platform and make it appear as any full-fledged **BIG IoT Platform**.
7.3. Interfaces

7.3.1. BIG IoT API

Programming Interface P1

The Programming Interface P1 is offered by the BIG IoT Provider Lib. It allows a developer to extend an IoT platform or service programmatically in order to authenticate it as a Provider on the BIG IoT Marketplace (M1 interface) and to register new offerings on the BIG IoT eXchange (M2 interface). The Provider Lib will also allow the programmer to register a callback function for each offering that is called when a Consumer requests access to the respective information or functions of the offering. Once a Provider instance is terminated, it should call the unregister call of the Provider Lib to unregister all the offerings associated with that Provider (M2 interface).

Programming Interface P2

The Programming Interface P2 is offered by the BIG IoT Consumer Lib. It allows a developer of a BIG IoT Application or Service to discover desired offerings programmatically on the BIG IoT Marketplace based on a semantic query (M3 interface). The BIG IoT eXchange returns the set of matching offerings, allowing the Consumer application or service logic to check the conditions (e.g. price and/or license) of the returned offerings in order to select those that match the applications or service needs. The Consumer can further subscribe to the selected offerings on the BIG IoT eXchange (M4 interface). Upon successful subscription, the Consumer will obtain the required credentials to access the offering on the Provider platform or service (A1 interface).

Programming Interface P3

The Programming Interface P3 is offered by the BIG IoT Proxy Lib. It allows a developer of a constrained device-level IoT platform, e.g. a platform that is not always available due to power-saving or a mobile access, to authenticate it as a Provider on the BIG IoT Marketplace (M1 interface) and to register new offerings on the BIG IoT eXchange (M2 interface). As such, the P3 interface provides the same functionality as the P1 interface when it comes to interactions with the BIG IoT Marketplace. With respect to providing access to the platform resources (via the A1 interface), the P3 interface however offers the option to push information that is locally available to a BIG IoT Proxy Service offered on a highly reliable and available infrastructure, such that a Consumer can obtain them from there at any time. With respect to tasking, the P3 interface offers a means for the constrained platform to pull all tasks...
from the proxy service that have been queued as the platform was in power saving mode or disconnected.

Marketplace Interface M1

The **M1 Interface** is the **BIG IoT Marketplace API** that is used by **Provider** and **Consumer** instances to authenticate themselves on the **BIG IoT Marketplace** at start-up. Upon an authentication request, the **BIG IoT IdM** checks the **Provider** or **Consumer** credentials (e.g. an API key and/or a certificate) and authorizes the instance. Upon success, it will return the required credentials to the **Provider** or **Consumer Libs** that are responsible for any further communication and interaction with the Marketplace.

Marketplace Interface M2

The **M2 Interface** is the **BIG IoT Marketplace API** that is used by **Provider** platforms or services to register offerings on the **BIG IoT Marketplace** programmatically. Once a registration request is received, the **BIG IoT eXchange** validates the offerings and stores them in a semantic database for later queries. The **BIG IoT eXchange** will assign a unique **OfferingID** and return this to the **Provider**. Alternatively, the developer or administrator of a **Provider** platform or service can also register offerings via the **Web Portal** on the Marketplace.

Subsequent registrations using the same **OfferingID** will indicate that the original offerings should be refreshed or updated. In this case, the registration request must only include the changed information elements.

The **M2 Interface** is also used by the **Providers** to unregister offerings upon termination of the platform or service, or simply the termination of the offering.

Marketplace Interface M3

The **M3 Interface** is the **BIG IoT Marketplace API** that is used by **Consumer** applications or services to discover offerings on the **BIG IoT Marketplace**. Once a discovery request is received, the **BIG IoT eXchange** validates the query and performs a search query on the semantic database. The **BIG IoT eXchange** will also consider the **Offering Access Control Lists** indicated in the offerings when compiling the list of matching offerings to be returned to the **Consumer**.

In the response, the **BIG IoT eXchange** will include all matching and validated offerings.

Note: In case the developer, user or administrator of a **Consumer** application or service uses the **Web Portal** to manually subscribe to offerings on the Marketplace, the discovery response will also include the unique Consumer credentials, indicating to the application or service which offerings have already been subscribed.
Marketplace Interface M4

The **M4 Interface** is the **BIG IoT Marketplace API** that is used by **Consumer** applications or services to subscribe to offerings on the **BIG IoT Marketplace**. Through the subscription the **Consumer** application or service explicitly confirms its consent regarding the offering’s license, price, etc. and indicates his intent to gain access to the offered resources. Once a subscription request is received, the **BIG IoT eXchange** provides the **Consumer** unique **credentials** to access this offering in the response message. In case the offering has expired or has been updated by the provider, the **BIG IoT eXchange** will deny the subscription and indicate the cause in the response.

Alternatively, to the subscription through the **Consumer** application or service, the developer, user or administrator of the **Consumer** can also subscribe to offerings via the **Web Portal** on the Marketplace, prior to running the application or service. In this case, the unique **Consumer credentials** will already be provided as part of the discovery response.

The **M4 Interface** is also used by the **Consumer** or Web Portal to unsubscribe from offerings, e.g. upon termination of an application or service.

Marketplace Interface M5

The **M5 Interface** is the **BIG IoT Marketplace API** that is used by the Consumer and Provider Libs to send accounting information to the **BIG IoT Marketplace**. Depending on the **Accounting Type** (e.g. per call, per message, per volume) defined in an offering description, both the Consumer and Provider Lib will perform the accounting on the **Access Interface A1**, and send those records in regular time intervals to the **BIG IoT Accounting Service**.

Marketplace Interface M6

The **M6 Interface** is the **BIG IoT Marketplace API** that is used by the Consumer Lib to report access service level measurement results and access failures to the **BIG IoT Marketplace**. The Consumer Lib measures quantitative metrics with respect to the service level of access an Offering, such as response times, throughput and connection failure rates on the provider end. The results of those measurements are provided to the marketplace in periodic reports. The marketplace in turn aggregates the results from different Consumers and computes an overall access service level for the Offering and Provider. Furthermore, in case the access to an offering fails repeatedly, the Consumer Lib reports this access failure immediately to the Marketplace, which then triggers a validation process. If the Offering is not available from the
marketplace, it will be deactivated to avoid that other consumers also attempt to use it without success. Once the Provider of such Offering is activated or connected again, it will re-register the Offering Description, and thus, re-activate the Offering.

**Access Interface A1**

The **A1 Interface** is the interface via which a **Consumer** gets access to resources offered by a **Provider**. The **BIG IoT Platform or Service** (as **Provider**) will offer the A1 interface and describe the means to access it in the offering.

In case of a **BIG IoT Service**, the A1 interface is provided by the **Provider Lib** that is used to connect the service into the **BIG IoT Ecosystem**. Depending on the **Provider Lib** implementation, it will support different access means. All **Provider Libs** support the access paradigm based on HTTP Request/Response. Optionally, a **Provider Lib** can also support other protocols (e.g. WebSocket, MQTT) or other access paradigms (e.g. streaming).

In case of a **BIG IoT Platform**, the A1 interface is either provided by the **Provider Lib** that is used to connect the IoT platform into the **BIG IoT Ecosystem** (Integration Mode 1/2/4), or by the platform itself (Integration Mode 3).

**Proxy Interface PR1**

The **PR1 Interface** is provided by the **BIG IoT Proxy Service** for **Integration Mode 4**. It offers a **constrained BIG IoT Platform** a means to:

1. store information (e.g. sensing or computed data) on the proxy, which in turn serves it to **Consumer** applications or services, on behalf of the constrained platform;
2. queue access requests to a function (e.g. a task) offered by the constrained platform on the proxy, such that it can process them upon wake-up or reconnection.

Note that due to the asynchronous nature of access to a constrained platform (2), these requests are limited to task-based functions, which can be handled by the platform at a later stage and without sending back any results. Related status information can of course also be provided by means of an information offering, that is pushed to the proxy service and then accessible at any time.

The **PR1 interface** will be based on a simple Web-based protocol (e.g. REST) in order to use this also from highly **constrained devices**.

TBD: Further details on this interface will be defined at a later stage.
7.3.2. Integration Interfaces

Integration Interface I1

The I1 Interface refers to the existing interface offered by an IoT platform that is integrated into the BIG IoT Ecosystem according to Integration Mode 2. The I1 interface is therefore outside the scope of BIG IoT.

The BIG IoT Gateway Service has to support and use the I1 interface in order to gain access to offerings provided by the integrated IoT platform upon a request from a Consumer application or service.

Integration Interface I2

The I2 Interface refers to the existing interface offered by an IoT platform that is integrated into the BIG IoT Ecosystem according to Integration Mode 3. The I2 interface is therefore outside the scope of BIG IoT.

In Integration Mode 3, access to the offered resources will be directly obtained on the integrated IoT platform. This implies that the A1 interface implementation on the Consumer Lib must directly support the I2 interface offered by the integrated IoT platform.

The BIG IoT Management Service may also use the I2 interface to gain access to the integrated IoT platform in order to obtain information about the resources to be offered on the marketplace. However, since the information on the offerings to be registered by the management service might also be provided through other means, the support of the I2 interface is optional.

7.3.3. Web Interface

The Web Portal of the BIG IoT Marketplace can be accessed via a normal Web API. It offers developers and administrators of the Provider or Consumer organizations access to create and manage offerings, queries and subscriptions via a normal Web browser. Moreover, developers can also gain access to BIG IoT related documentation, sample code, as well as download the BIG IoT Libs via this Web interface.
7.4. Semantic Models and Framework

7.4.1. BIG IoT Semantic Models

![BIG IoT Semantic Modelling Layers Diagram]

7.4.1.1. BIG IoT Semantic Core Model

The semantic core model defines the vocabulary and structure required to describe Offerings and OfferingQueries in the context of BIG IoT.

One of the key aspect when designing a semantic model is re-using of knowledge. Once a semantic model is created for a domain, it should be (at least to some degree) reusable for other applications in the same domain. To simplify both semantic model development and reuse, a modular design is beneficial. Based on the project specification and the domain model in section 5.1.2, the semantic models can be modularized according to their scope, as follow:

- Organization module
- Provider module
- Offering module
- Consumer module
- Query module

Key functional concepts in this model are the *OfferingCategory*, *Data*, and *Endpoints*. The OfferingCategory is used to classify the Offerings on a marketplace. The semantic core model defines among other things, the classes "core:Offering", "core:OfferingCategory" and
"core:Data". The class "core:OfferingCategory" is used to model the category to which an offering belongs to. The class "core:Data" is used to model the Input and Output Data of an Offering.

The details of the BIG IoT Semantic Core Model are described in Deliverable D3.2.b.

7.4.1.2. BIG IoT Semantic Domain Models

While the semantic core model defines the core vocabulary and concepts, the Offering Descriptions (ODs) and Offering Queries also require domain independent and domain dependent vocabulary to describe the offerings. Therefore, domain dependent and independent semantic models are used to annotate the OfferingCategory and Input/Output Data of an Offering Description. Along with offering categorization and data modelling, the semantic domain model also defines the vocabulary to semantically annotate the domain dependent features of an offering.

At the moment, the BIG IoT Marketplace supports two domain models:

- Mobility (Parking, Traffic, Bus, Charging,..)
- Environment (Air Quality, Pollution, ..)

For example, the Mobility Domain Model defines vocabulary to semantically annotate "core:OfferingCategory", "core:Data" and metadata specific to mobility related Offerings. Also, the class "mobility:ParkingSpace" can be used to annotate a parking space and its features such as a parking space name or a parking space location.

The details of those semantic domain models are described in Deliverable D4.2.b.

7.4.1.3. BIG IoT Semantic Application Model

To create Offering Descriptions both the core model and the domain models are used. The semantic application model is created to connect the semantic core model with domain independent and domain dependent models. As such, the semantic application model establishes the relationship between the core model and domain models. An example is shown in Figure 9.
The details of the BIG IoT Semantic Application Model are described in Deliverable D4.2.b [25].

7.4.2. Semantic Storage, Interfaces and Frameworks

As part of the Marketplace persistence framework and back-end layer, we needed to deploy a semantic database (i.e. a triple store) that is able to store all semantically annotated (meta)data in a way that supports semantic discovery.

This database represents the main repository where all semantic (meta)data are stored. The following subsections will provide more details about the semantic components and its association with other parts of the marketplace.

7.4.2.1. Triple Store

In the persistent database we will need to store all the (meta)data created and collected by marketplace customers, e.g. the information about Providers, Consumers, Offering Descriptions and Recipes. In our research for a semantic interoperability in BIG IoT we decided to model our Providers, Consumers, Offering Descriptions and Recipes following an RDF schema model, annotated with our BIG IoT Semantic Core Model.

Due to the nature of such kind of (meta)data we need to choose the best solution for storing, managing, accessing and retrieving information. The semantic descriptions are generated following the BIG IoT Semantic Core Model, annotated according to the BIG...
**IoT Semantic Domain Models**, and mapped with the **BIG IoT Semantic Application Model** vocabularies and then loaded into the RDF triple store. A RDF triple store is a type of graph database that stores semantic facts. As a graph database, a triple store stores semantic data as a network of objects with materialized links between them. This makes a RDF triple store a preferred choice for managing highly interconnected data. Triple stores are more flexible and less costly than a relational database, for example.

Such a semantic graph database is also capable of handling powerful semantic queries and using inference for uncovering new information out of existing relations. In contrast to other types of graph databases, RDF triple stores support the concurrent storage of data, metadata and schema models (e.g. the so-called ontologies). Models/Ontologies allow for formal description of the data. They specify both object classes and relationship properties, and their hierarchical order. Our BIG IoT semantic models describe this for our resources.

This allows to create a unified knowledge base grounded on common semantic models that allows to combine metadata coming from different sources making them semantically interoperable to

- create coherent queries independently from the source, format, date, time, provider, etc.
- enable the implementation of more efficient semantic querying features
- enrich the data by making it more complete, more reliable and more accessible
- enable inference as triple materialization from some of the relations

The details of the BIG IoT eXchange Persistence Framework is described in Deliverable D4.1.b [24].

**7.4.2.2. GraphQL and Semantic Framework Integration**

According to our high-level architecture the BIG IoT Marketplace API aggregates interfaces, collecting data from internal services (e.g. Access Management, eXchange, Accounting) and provides it to clients like the Web Portal or the Provider and Consumer Libs. For example, when a user creates an Offering via the Web Portal, it fetches all necessary data to inform the user about all the options (e.g. OfferingCategories and the associated semantic concepts to annotate the Input and Output Data).

Considering that the Web Portal UI needs a simple query language to retrieve the data used to populate the graphical user interface, it was decided to utilize an API interface based on GraphQL, a query language for APIs, that has proven to be a very easy to implement, simple
to query the necessary data, and extensible. However, considering the need to persist all the (meta)data in a semantic database (i.e. a triple store) in order to leveraging the power and functionalities of semantic queries we need to implement a SPARQL endpoint on top of it.

Once the data is expressed in RDF, the use of a SPARQL (as query language and engine) to find specific knowledge information is straightforward. These queries can go from simple to more complex and select whole documents or sub-patterns that match conditions defined on data content and structure.

In order to integrate the two query interfaces, GraphQL and SPARQL, we established a way to translate and map queries between GraphQL and SPARQL. Figure 10 illustrates how we integrate GraphQL into our BIG IoT Semantic Framework.

![FIG. 10: Architectural details on the Integration of GraphQL and the BIG IoT Semantic Framework](image)

In the following part, we present the actual development implementation of this mapping and translation process.

**GraphQL <> Semantic Serializer**

This component maps GraphQL documents to RDF. The former are represented as Scala [26] objects and the latter as Jena [27] models (an encapsulation of RDF graphs in Java objects). As mentioned in Deliverable D3.2.b [23], the translation could be done automatically, in theory. However, because of the limited support to manipulate Scala objects from Java, our implementation has to perform the translation in an ad-hoc manner: a serialization class having a static method for all entities of the BIG IoT Core Model does this translation.
Semantic Reasoner

In this section, we introduce the BIG IoT Semantic Reasoner. This component is a rule-based inference engine, where the rules are generated and used to inference new knowledge that is stored in the triple store. We are currently using a Jena inference subsystem to derive additional information about the offering, which is entailed from the Offering Description during registration together with any optional ontology information, and the axioms and rules associated with the reasoner. For example, the reasoner is fed with the semantic models (Application, Domains, ...) and via ontology reasoning it uses the rules to infer the relationships between categories of offerings and related input/output data type and the resulting inferred knowledge is uploaded in the triple store. The inferred knowledge and the relationships that are now part of the overall semantic model, which is used by the "Semantic Entity Creation, Update and Query Engine" component for the following use-cases:

1. Offering Discovery: The offering discovery process requires the reasoner to infer the relationship between the category and its subcategories. For example, if a Consumer wants to discover all parking related offerings, the offerings with sub-categories of parking (e.g. parking site management or parking space management) need also be included in the discovery response. Below we show an example for a reasoning rule to infer all the possible parent categories in the category hierarchy tree that can be associated with the offering. This process is executed when the offering is registered to the marketplace.

   \[
   \text{Reasoning rule to infer the relationship between the offering and its category} \\
   @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> \\
   @prefix core: <http://schema.big-iot.org/core/> \\
   @prefix skos: <http://www.w3.org/2004/02/skos/core#> \\
   @prefix schema: <http://schema.org/> \\
   (\(?offering schema:category \?category\), \\
    \(?parentCategory skos:narrower \?category\) \\
    \rightarrow \(?offering schema:category \?parentCategory\) .
   \]

2. Offering Description: When a Provider defines an Offering Description via the Web Portal UI, inferred knowledge determines, depending on the selected offering category and sub-categories, the applicable Input and Output Data types. The Web Portal UI uses these types to assist the user in describing the offering according to our semantic application models. The
following rule is used during start-up of the marketplace to derive such knowledge based on the BIG IoT semantic models.

*Reasoning rule for category and input/output data*

```rdfs
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
@prefix core: <http://schema.big-iot.org/core/>
@prefix skos: <http://www.w3.org/2004/02/skos/core#>
@prefix schema: <http://schema.org/>

(?category skos:narrower ?subcategory),
(?category rdf:type core:OfferingCategory)
  -> (?subcategory rdf:type core:OfferingCategory) .

(?category rdf:type core:OfferingCategory),
(?category core:refersTo ?class),
(?prop schema:domainIncludes ?class)
  -> (?prop rdf:type core:DatatypeAnnotation),
      (?category core:expectedAnnotation ?prop) .

(?category core:expectedAnnotation ?prop),
(?prop rdf:type core:DatatypeAnnotation),
(?prop schema:rangeIncludes ?class),
(?otherprop schema:domainIncludes ?class)
  -> (?otherprop rdf:type core:DatatypeAnnotation),
      (?category core:expectedAnnotation ?otherprop) .

(?supercategory skos:narrower ?category),
(?category rdf:type core:OfferingCategory),
(?category core:expectedAnnotation ?prop)
  -> (?supercategory core:expectedAnnotation ?prop) .
```
Semantic Entity Creation, Update and Query Engine

As described above, for some simple offering queries, the Portal UI uses GraphQL to retrieve the data used to populate the graphical user interface. However, for more complex queries that require reasoning we use the triple store. Therefore, to query and retrieve the data, we use the SPARQL query language to manipulate the RDF data. In this component, we also manage the creation and manipulation of the semantic entities to be stored in the triple store. For example, the semantic entities and their relationships that form the offerings descriptions are created/updated via the SPARQL commands: INSERT and UPDATE, and the information are retrieved from the store via SPARQL CONSTRUCT and SELECT queries. This example SPARQL query retrieves the tree of all the offering categories.

Get all categories

```
PREFIX schema:      <http://schema.org/>
PREFIX core:        <http://schema.big-iot.org/core/>
PREFIX skos:        <http://www.w3.org/2004/02/skos/core#>
PREFIX rdf:         <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs:        <http://www.w3.org/2000/01/rdf-schema#>
PREFIX mobility:    <http://schema.big-iot.org/mobility/>
PREFIX owl:         <https://www.w3.org/2002/07/owl#>
PREFIX td:          <http://www.w3c.org/wot/td#>
PREFIX xsd:         <http://www.w3.org/2001/XMLSchema#>

CONSTRUCT {
    ?category a core:OfferingCategory;
    rdfs:label ?label;
    skos:narrower ?sub.
}
FROM <http://big-iot.eu/ontologies#>
WHERE {
    ?category a core:OfferingCategory.
    OPTIONAL {
    }
}
```
7.5. Runtime View

The following diagrams display the message sequence diagrams for registering offerings (M2 Interface), and discovering (M3 Interface) and subscribing (M4 Interface) to offerings on the marketplace, as well as accessing the resources on the Offering Provider (A1 Interface).

The involved stakeholders are i.) BIG IoT Application, Service and Platform Providers (i.e. the responsible users or developers) that interact with the marketplace to either offer or discover resources, ii.) Offering Provider and Consumer instances, and iii.) the Marketplace Portal and eXchange.

7.5.1. Offering Registration

Figure 11 describes the execution flow to authenticate on the Marketplace (M1) and register offerings on the eXchange (M2). Upon registration of an offering, it is made known on the marketplace, allowing consumers to find it. It should be noted that offerings on the eXchange can also be stored in inactive state. This means that the offering provider has to activate the offering first.

To register an offering, the service or platform provider logs in via the marketplace portal. The portal authenticates the provider (organization) on the marketplace and returns the required access token for further API calls to the marketplace upon successful authentication. Note also that the portal will allow service/platform providers to create new offering provider and consumer instances. During this step, the marketplace will create dedicated API keys/secrets for each consumer or provider instance. Service/platform providers (or their developers) will be able to obtain those API keys/secrets to use them within the respective software components.
The platform provider then uses the graphical user interface provided by the portal to describe the resources to be offered via the marketplace. This includes a definition of the offering name, the semantic category (and sub-category) of the offering, the input and output data associated with the access request, as well as metadata such as the region, where the offering originate, the license type, the price, etc.

In static/manual mode (Mode 1), the platform provider registers the offering description through a button click on the GUI, which issues the offering registration API call (M2) to the marketplace. The eXchange in turn stores the offering as inactive. A unique **OfferingID** is assigned and shown to the user in order to refer to this offering in API calls at a later stage.
In dynamic/programmatic mode (Mode 2), i.e. when the offering registration should be performed at run-time because of dynamic information, as for instance the location or a price is used, the developer may use the graphically generated offering description as a template for creating the offering description in the source code. This mode is used in case a service or platform needs to finalise the offering description as part of the program logic, based on information that are only available at run-time. In this case, the offering description obtained from the GUI on the Web Portal will serve as template, and the dynamic information are filled in at run-time.

When the service/platform provider starts up the offering provider, the software instance first authenticates itself on the marketplace in order to obtain the required access token for further API calls to the marketplace. In mode 1, the offering provider then triggers the activation of the previously stored offering on the eXchange. In mode 2, the offering provider finalizes the offering descriptions at run-time, and then registers and activates them directly on the eXchange.

After these steps are performed, the offering is "active on the marketplace" and can be discovered by consumers.

7.5.2. Offering Discovery and Subscription

Figure 12 describes the discovery (M3) and subscription (M4) to offerings on the marketplace. To discover and subscribe to offerings, the application or service provider logs in through the marketplace portal as already described in the previous section.

Afterwards the developer can define an offering query through the GUI provided by the Web Portal. Here developers can define the semantic (sub-)category of the offering they are looking for, the required input and output data, as well as the region, license, price, etc. of interest. The query description is then submitted to the marketplace, where it is stored. A unique QueryID is created that allows a consumer to refer to the specific query in API calls at a later stage. The eXchange also runs the query at this point in time in order to see which of the currently available offerings would match. The matching offerings are returned and displayed.
In static/manual mode (Mode 1), the user will then manually select the offerings of interest and subscribe to them via the portal GUI, which triggers the portal to transmit the IDs of the subscribed offerings to the marketplace.

In dynamic/programmatic mode (Mode 2), the developer either uses the graphically generated query description as a template for creating the query in software, or simply uses the QueryID in case the query does not need to be altered during the lifetime of the consumer application or service. Important however is the fact that subscription to offerings is performed dynamically. This allows the consumer instances to discover and subscribe to new resource offerings at run-time, as soon as they are offered on the marketplace. No manual subscription via the Marketplace portal is required. In this mode it is therefore required that
the consumer logic either has the necessary policies in place to auto-subscribe to new matching offerings, or it may involve the user of the application or service at run-time (as part of the application or service logic).

When the provider starts up an offering consumer, the software instance first authenticates itself on the marketplace in order to obtain the required access token for further API calls to the marketplace.

In mode 1, an offering consumer then submits a discovery request based on the previously stored query on the eXchange. In turn, the eXchange returns a list of all matching and already subscribed offerings, together with the corresponding Offering Access Tokens (specific for the consumer), which are needed for accessing the resources on the provider end. This is described in detail in the following section (see Section 7.4.3).

In mode 2, the offering consumer instance may finalize the query at run-time (i.e. programmatically), allowing it to augment or extend the description with dynamic information (such as location), or simply uses the QueryID of a previously defined query on the Marketplace. The consumer instance sends the query description or QueryID to the eXchange as part of the discovery request. In this case, the eXchange returns a list of all matching offerings at this point in time. The application or service logic will then select either based on local policies or by interacting with the end user, which offerings to subscribe to. Upon the subscription call to the marketplace, the eXchange returns the Offering Access Tokens required by the consumer to access the resources on the provider end.

After these steps are performed, the consumer has discovered offerings of interests and subscribed to them on the marketplace, and is now in a position to access the desired resources.

7.5.3. Resource Access

Figure 13 describes how a consumer accesses the offered resources on the provider end (A1). The offering consumer can either access data/information or execute functions on the offering provider’s side. In the access request to the provider (via the Access Interface A1), the consumer uses the specific offering access token, which authorizes the consumer to access the offered resources, as well as the input data, which can be used by the provider to filter the output data to the precise set of data records that are of interest to the consumer. The offering provider, upon receiving an access request, validates the offering access token, by checking the Marketplace signature and subscription ID. If the token is valid, the data/information to be transmitted to the consumer is delivered, or the respective function call is executed.
Once the data exchange is ongoing, both the offering consumer and the offering provider issue accounting records to the marketplace via the M5 Interface. According to the access type (request/response vs. streaming), they send information regarding the consumption of offerings, e.g. number or requests, number of bytes transferred, number of data records offered, streaming session time, etc. The marketplace will use this information as a basis for charging and billing. Note that in case the marketplace obtains conflicting accounting reports from the Consumer and Provider, it will inform the Consumer and Provider during the next interaction with the marketplace. Also the offering access token, which typically as a very short (in the order of minutes) lifetime will not be refreshed.

FIG. 13: Message sequence flow for accessing the resources of an offering
Besides the accounting information, the Consumer will also send reports about the perceived service level from the given Provider. These reports include information regarding the response time, jitter, throughput, and failure rates. The marketplace will use those information to rate Providers and Offerings, allowing Consumers to also select competing Offerings based on such information.
7.6. Deployment View

The following figure illustrates an exemplary deployment scenario of the various BIG IoT components in a smart city context. This example scenario assumes that the city council is interested in promoting smart services for the city in order to improve the quality of life for the citizens. To do so, the city council decided to host a dedicated BIG IoT Marketplace on its Cloud/IT infrastructure to facilitate organisations, SMEs, corporations in the city to discover and exchange IoT resources, and thus to promote the creation of smart applications and services for the city, to overcome today's cross-platform, cross-domain and cross-organisational hurdles.

In this example scenario, we have the following stakeholders:

- **the City Council**, who hosts and runs the BIG IoT Marketplace on its Cloud/IT infrastructure, and offers various city related data resources on the marketplace, namely the data of parking sensing cameras and air quality sensors that are deployed around the city. The parking information captured by the device-level parking camera platforms are offered on the marketplace and can be directly accessed by services that require such data. In contrast, the air quality sensor data are offered through a BIG IoT Proxy Service, hosted by the city council, as the device-level sensing platform is not always accessible due to energy saving. Furthermore, the city council performs environmental monitoring around the city in order to warn its citizens when pollution levels (e.g. fine dust) reach critical levels and restrict traffic flows in the affected areas. For this, the responsible department uses an environmental monitoring application that is BIG IoT enabled, and thus allows discovering and taking into account all available data sources that offer relevant information.

- **Parking Lot Providers**, who operate commercial parking lots and offer information about their parking lot on the marketplace. In this example, they extend their local parking lot management platform by leveraging the BIG IoT Lib/SDK. Their platforms in turn become full-fledged BIG IoT Platforms that are capable of offering parking related information on the marketplace and serving them via the access interface to interested consumers.

- **a Parking Data Provider**, who owns a parking lot sensing infrastructure around the city (e.g. for street parking lots) and offers those data to the city and other interested service providers. Those sensors are already deployed and connected to a legacy sensing platform, which cannot be modified or extended. In order to allow the provider to offer those parking data also on the BIG IoT Marketplace, a BIG IoT Gateway
Service is deployed on the provider's Cloud/IT infrastructure that is responsible to register the respective parking information offerings on the marketplace and translate access requests from interested BIG IoT consumers to the proprietary interface of the legacy platform.

- a **Smart Parking Service Provider**, who hosts and runs a smart parking service for the city. The service uses the BIG IoT Marketplace to discover all available parking related data/information sources around the city (i.e. from the city council, the parking lot providers and the parking data provider) and uses them to offer "higher-level" smart parking information covering the whole city and all available parking lot types to the end-users (e.g. car drivers). This provider also offers its smart parking service on the Marketplace, such that BIG IoT Applications can discover them at run-time, allowing in-car or smartphone apps of visitors to the city to also utilize the service.

- a **Environmental Monitoring Service Provider**, who collects, combines and integrates various environmental data sources (e.g. air quality data, fine dust data) across the city. One the one hand, the provider uses in-car air quality sensing devices, which have been offered to selected car owners, in order to collect own data across the city. Since those sensors use the driver's mobile connection to send the data, the service provider deployed a BIG IoT Proxy Service on its Cloud/IT infrastructure, allowing the sensor platform to efficiently communicate and store the sensing data on the backend. On the other hand, the provider combines its own data with air quality data offered by the city council in order to compute citywide air quality maps. This environmental information are offered again on the marketplace as basis for various applications and services (e.g. the green routing service).

- a **Green Routing Service Provider**, who uses environmental information of the city to compute environmental-friendly routes with the aim to reduce pollution levels in areas where critical levels have been reached for car or truck drivers, and to recommend healthy routes with high air quality levels to cyclists. The green routing service uses the marketplace to discover available environmental information sources and uses them as a basis for the route computation. The green routing service is also offered on the BIG IoT Marketplace, allowing BIG IoT Applications of visitors to the city to also discover and utilize the service.

- **Car Drivers**, who use the in-car or smartphone apps for parking and green routing. The fact that these apps are implemented as BIG IoT Applications, which implies that they use the marketplace to discover suitable information sources and services at run-time, ensures that these apps also work across city boundaries. In particular,
these apps are able to discover and use local information sources and services depending on the car’s location.

- **Bicycle Riders**, who use a smartphone app showing an air quality map of the surrounding in order to choose a healthy ride. The BIG IoT Application uses the marketplace to discover relevant local air quality information sources, such as the environmental monitoring service.

FIG. 14: An exemplary deployment scenario of the BIG IoT architecture components in a smart city context

Note: For better readability, a bigger illustration of this smart city deployment scenario is also provided in Annex 2 of this deliverable.

As indicated above, Figure 14 illustrates merely an exemplary deployment scenario in a smart city context. Ongoing work as part of the WP5 defines the concrete deployment scenarios for the target pilot use cases defined in Deliverable D2.2.a [20].
8. Proof-of-Concept Implementation and Demonstrator

This section presents the demonstrator of our BIG IoT architecture components. In an end-to-end scenario, the practicability of the BIG IoT architecture 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 demonstration scenario highlights that the discovery and integration of IoT resources provided by heterogeneous platforms and various organizations becomes possible through the developed API and the Marketplace. The key architecture components demonstrated are (1) the BIG IoT Marketplace (i.e. the Web Portal, the eXchange, the Tripe Store), (2) a Smart City Dashboard application (a BIG IoT Application as Consumer), (3) a Parking Space Information Provider connected to the OpenIoT platform (a BIG IoT Gateway Service as Provider), (4) a second Parking Space Information Provider in Barcelona offering parking space data (a BIG IoT Service as Provider), and (5) an Air Quality Information Provider offering data from the BEZIRK device-level platform installed in SEAT cars (a BIG IoT Proxy Service as Provider).

Figure 15 shows the overall demonstration scenario, including the stakeholders (in the middle), the applications and data sources (on the right), as well as the BIG IoT Marketplace, and Providers and Consumers using the BIG IoT SDKs, i.e. Consumer and Provider Libs (on the left).

The sequence of interactions in this demonstration scenario are numbered in the figure and described in the following:
1. The developer of the OpenIoT Parking Space Information Provider visits the BIG IoT Marketplace Web Portal. After signing in, she creates a new organization for her company and a new Provider instance. Guided by the Web Portal, she then creates a first Offering. The GUI recommends supported OfferingCategories. Once she selects the category and possible sub-categories (i.e. parking) for the new Offering, the GUI recommends suitable Input and Output Data properties along with their semantic annotations. The developer also specifies the license type of the offered data, as well as the charging model (e.g. per access) and price (e.g. 0.02 Euro). Once ready, the Offering Description is stored on the Marketplace. Figure 16 illustrates the GUI of the Marketplace to create new offerings.

2. In order to register this Offering on the Marketplace, and activate it such that Consumers are able to discover and access it, the developer has to extend its Parking Management Platform using the BIG IoT Provider Lib. The Provider Lib offers methods calls for creating a new Provider instance, authenticating this on the Marketplace, and to create Offering Description and register them. The code snippet illustrated in Figure 17 shows how the Provider Lib is used and what a developer needs to implement to make its platform or service integrate with BIG IoT.
details on how to use the BIG IoT Libs are described online in our Developer Guide [22].

```java
ProviderSpark provider = new ProviderSpark(PROVIDER_ID, MARKETPLACE_URI, PROVIDER_ADDRESS, PROVIDER_PORT);

// Authenticate provider on the marketplace
provider.authenticate(PROVIDER_SECRET);

// Construct Offering Description based on the one stored on the Marketplace
RegistrableOfferingDescription offeringDescription = provider.createOfferingDescription(OFFERING_ID)
    .useOfferingDescription(OFFERING_ID)
    .withProtocol(endpointType, HTTP_GET)
    .withRoute("parkingInfo")
    .withAccessRequestHandler(accessCallbackParkingInfos);

// Register a new Offering on the Marketplace and bring up the Endpoint
provider.register(offeringDescription);

private static AccessRequestHandler accessCallbackParkingInfos = new AccessRequestHandler()
{
    @Override
    public BigIoTHttpResponse processRequestHandler (OfferingDescription offeringDescription, Map<String, Object> inputData) {
        JSONObject results = ...
        return BigIoTHttpResponse.okay().withBody(results.toJSONString()).asJsonType();
    }
};
```

FIG. 17: Java Source code extract of Demo Parking Space Provider using the Provider Lib

3. When the developer starts up the BIG-IoT-enabled Parking Information Service, the new Provider will authenticate itself on the Marketplace (based on the Provider ID and secret obtained in step 2), and register the Offering Description. Once an Offering Description is registered on the Marketplace, the Offering can be discovered and accessed by a Consumer.

4. The developer of the Air Quality Information Service, who also wants to monetize her air quality data on the BIG IoT Marketplace, visits the Marketplace, and first creates a new organization for her company and a new Provider instance for the new data source. She also creates a new Offering following the guidance of the Web Portal - similar to step 1.

5. The developer then extends its service by using the Provider Lib, following the example provided in step 2.

6. When the BIG-IoT-enabled Air Quality Information Service is finally started up, it will - like in step 3 - register the new Offering and wait for Consumers to access the data.

7. The developer of the Smart City Dashboard also visits the BIG IoT Marketplace in order to find suitable data sources for the application. She can browse and search
through existing offerings to get an idea what type of data are available on the Marketplace. The developer then creates two Offering Queries to describe the desired information for the Smart City Dashboard (as a Consumer): one for parking space information and one for air quality data. The queries can also include metadata like, for example, the accepted license types and a maximum price.

8. Once the queries are created, the developer implements or extends her dashboard application using the BIG IoT Consumer Lib. The Consumer Lib offers methods calls for creating a new Consumer instance, authenticating this on the Marketplace, and to discover Offerings (based on Offering Queries) and subscribe to them.

9. When the developer of the Smart City Dashboard application starts it up, the new Consumer will authenticate itself on the Marketplace (based on the Consumer ID and secret obtained in step 7), and run a discovery for the desired offerings (based on the Offering Queries created in step 8). In this step of the demo one matching parking information offering and one matching air quality information offering will be discovered. The Consumer application then checks the obtained offering descriptions, to make sure they fulfil the application policies with respect to the licensing term, the price, etc. If the discovered Offerings match the needs of the consumer application, it subscribes to them on the Marketplace. During this step, the Consumer will also obtain the necessary security credentials (Offering Access Tokens) to later access the offering.

10. Once subscribed, the Consumer can call the access method to get the information of the subscribed Offering and then use it in the application logic. In this step, the Consumer accesses the data offered by the OpenIoT Parking Space Information Provider and visualizes it on the Smart City Dashboard. It is important to note here that the Consumer did not have to be adapted/extended to call the newly discovered Provider and implement a new API to access the data. Nor was it necessary to manually obtain an access token for the new data provider. The BIG IoT API with its semantic approach enables the Consumer to access various offerings (with different data) from different Providers using the same API.

11. Similar to step 10, the Consumer also accesses the data offered by the Air Quality Information Service and integrate them into the application. Figure 18 shows a screenshot of the Smart City Dashboard once both Offerings have been integrated.
12. In this step, another developer from a different organization also visits the Marketplace in order to create a second Parking Information Offering, like in step 1.

13. Following this, the developer extends her parking space management system using the Provider Lib as described in step 2.

14. Once the new system has also been BIG-IoT-enabled, it will authenticate the new Provider on the Marketplace during start-up and register the new offering. From that point in time, Consumers with matching Offering Queries will be able to discover the new offering.

15. Since the new Parking Information Offering also matches the Offering Query of the Smart City Dashboard Consumer (i.e. the region, the semantic category, the input and output data as well as the price, license, ...), the Consumer is informed about the new matching Offering. The consumer application subscribes then the new offering - as all conditions match the local policies.

16. Upon subscription to the new Parking Information Offering, the dashboard will access the new data source and integrate the data at run-time. As a result, the Dashboard visualizes the new parking spot information immediately, without a need to extend, update and restart the application. Figure 19 shows the screen shot including the added information.
FIG. 19: Smart City Dashboard after discovery, subscription and integration of a newly created Parking Information Offering

The demonstrator shows that the BIG IoT architecture with its API and Marketplace 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, and by different 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 operate on different scales (cloud-level vs. device-level platforms); and (3) addressing the evolvability problem of IoT applications and services, who rely mostly on manual integration of emerging IoT resource providers (e.g., new data sources).

Note that although this particular demonstration scenario incorporates only platforms from the smart city domain, the BIG IoT API and Marketplace can be utilized in other domains as well. The validation of the BIG IoT architecture for other use case scenarios and across domains is ongoing work by the project, aiming to address the five interoperability patterns identified in [1]. In particular, the project will demonstrate by means of three pilots (see Deliverables D2.2.b, D5.1.a and D5.3a for a detailed description of the target use cases) that the BIG IoT architecture is able to bridge the interoperability gap across heterogeneous IoT platforms, services and applications.
9. Conclusions and Outlook

This deliverable presents the results of the BIG IoT architecture work, which has been carried out from M3-M11 (Release 1) and M16-M19 (Release 2) of the project. The work was carried out by Task 2.4, but also involved experts from other project activities, in particular Task 2.1 (Analysis of technology readiness), Task 2.2 (Use Case definition), Task 2.3 (Requirements analysis and specification).

In section 1, the document introduced the project goals and challenges. Section 2 and 3 clarified the terminology that has been established by the project and the scope of the BIG IoT architecture work. Section 4 introduced the guiding use cases and requirements. In section 5 and 6, we explained the methodology and approach followed so far, as well as the core concepts and high-level design decisions that influenced the architecture. Section 6 also provides a high-level description of the BIG IoT architecture. In section 7, at the heart of the document, we presented the status of our BIG IoT architecture (Release 2) by providing i.) a view and description of the key building blocks as well as their interfaces, ii.) high-level message sequence charts for the key interactions, as well as iii.) a deployment view based on an exemplary Smart City scenario. Finally, in Section 8 we presented a demonstration scenario that has been implemented and demonstrated based on our Release 1 implementation 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, consisting of the BIG IoT API and Marketplace, 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 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 a "meeting place" 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. For this, we re-use existing semantic vocabularies, or establish missing ones, for relevant application domains 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 Release 2 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, and the results will be documented in Deliverables D3.1.b and D4.1.b respectively. To ground these specifications in public standards, we are actively participating to the W3C Web of Things group.

Finally, in order to ensure that our implementation efforts on the BIG IoT SDK (i.e. Provider and Consumer Libs) as well as the BIG IoT Marketplace is open and can be re-used or extended by the community, we are currently in the process of establish an open source project at Eclipse. The initial project proposal has already been approved by the Eclipse board. We are currently in the process of preparing the initial code contribution based on our Release 1 implementation of the BIG IoT API and Marketplace.
References


[22] BIG IoT Developer Guide, available online at: https://big-iot.github.io/


[27] Apache Jena, “A free and open source Java framework for building Semantic Web and Linked Data applications.”, available online at: https://jena.apache.org/
ANNEX I

BIG IoT Architecture Figure.
ANNEX II

Exemplary Deployment View of the BIG IoT Architecture Components.
DELIVERABLE 2.4.A – HIGH LEVEL ARCHITECTURE SPECIFICATION

Diagram showing various components and services related to a smart city infrastructure, including Parking Sensing, Smart City Cloud / IT Infrastructure, Air Quality Sensors, City Council Parking Camera Development, Parking Lot Provider IT Infrastructure, City Council PC Environmental Monitoring, and Bicycle services. The diagram uses symbols for BIG IoT Application, BIG IoT Gateway Service, BIG IoT Proxy Service, BIG IoT Service, BIG IoT Platform, and BIG IoT Marketplace API (control plane) and BIG IoT Access API (data plane).