D3.3 Context-Sensitive Security, Privacy Management, Adaptation Framework v1

<table>
<thead>
<tr>
<th><strong>Project Acronym:</strong></th>
<th>bloTope</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Project title:</strong></td>
<td>Building an IoT Open Innovation Ecosystem for Connected Smart Objects</td>
</tr>
<tr>
<td><strong>Grant Agreement No.</strong></td>
<td>688203</td>
</tr>
<tr>
<td><strong>Website:</strong></td>
<td><a href="http://www.bloTope-project.org">www.bloTope-project.org</a></td>
</tr>
<tr>
<td><strong>Version:</strong></td>
<td>1.0</td>
</tr>
<tr>
<td><strong>Date:</strong></td>
<td>31 March 2017</td>
</tr>
<tr>
<td><strong>Responsible Partner:</strong></td>
<td>UL</td>
</tr>
<tr>
<td><strong>Contributing Partners:</strong></td>
<td>UL, Aalto</td>
</tr>
<tr>
<td><strong>Dissemination Level:</strong></td>
<td>Public</td>
</tr>
</tbody>
</table>

Confidential – only consortium members and European Commission Services
## Revision History

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Author</th>
<th>Organization</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>03/01/2017</td>
<td>Jérémy ROBERT</td>
<td>UL</td>
<td>Initial draft</td>
</tr>
<tr>
<td>0.2</td>
<td>24/02/2017</td>
<td>Sylvain Kubler</td>
<td>UL</td>
<td>Revision 24 Feb. 2017</td>
</tr>
<tr>
<td>0.3</td>
<td>28/02/2017</td>
<td>Jérémy ROBERT</td>
<td>UL</td>
<td>Consolidated version</td>
</tr>
<tr>
<td>0.4</td>
<td>08/03/2017</td>
<td>Jérémy ROBERT</td>
<td>UL</td>
<td>Integration of the Holonix’s comments (Simone Parrotta)</td>
</tr>
<tr>
<td>0.5</td>
<td>16/03/2017</td>
<td>Jérémy ROBERT</td>
<td>UL</td>
<td>Integration of IS-Pratice’s comments (Hugo Kerschot)</td>
</tr>
<tr>
<td>0.6</td>
<td>20/03/2017</td>
<td>Jérémy ROBERT</td>
<td>UL</td>
<td>Integration of Itrust’s comments (Jean Lancrenon/Sankalp Ghatpande)</td>
</tr>
<tr>
<td>1.0</td>
<td>23/03/2017</td>
<td>Jérémy ROBERT</td>
<td>UL</td>
<td>Preparation of final version</td>
</tr>
</tbody>
</table>
Table of Contents

1. Introduction ........................................................................................................................................................................ 7
2. O-MI reference implementation ........................................................................................................................................ 9
   2.1. Reminder on the current reference implementation .................................................................................................. 9
   2.2. Discussion about the security in the reference implementation .................................................................................... 13
3. Security modules for the O-MI node ............................................................................................................................... 14
   3.1. Access control and authentication mechanism for the end-users of the O-MI webclient ............................................ 14
   3.2. Secure O-MI/O-DF information exchange for third party applications ................................................................. 17
4. Security & Privacy with the service catalogue IoTBnB .................................................................................................... 19
   4.1. Security management with IoTBnB .................................................................................................................................. 19
   4.2. Privacy management strategies with IoTBnB .................................................................................................................. 24
5. Conclusion ......................................................................................................................................................................... 27
6. References .......................................................................................................................................................................... 28
List of Tables

Table 2-1 Main Messaging Interfaces specified in the O-MI standards ................................................................. 10
Table 3-1 Parameters for asking the token .................................................................................................................. 17

List of Figures

Figure 1-1 bIoTope architectural framework .............................................................................................................. 8
Figure 2-1 O-MI reference implementation ................................................................................................................... 9
Figure 2-2 Illustration of the O-DF structure ................................................................................................................ 10
Figure 2-3 Webpage of the O-MI webclient (here for the node’s URL http://biotope.sntiotlab.lu:8080).............. 10
Figure 2-4 Discovering of the Object Netatmo in the Object(s) SnT and IoTlab ...................................................... 11
Figure 2-5 O-MI webclient UI after clicking ‘Read All’ (with the published O-DF tree), and selecting the InfoItem Temperature Indoor in the Object(s) SnT, IoTlab and Netatmo ..................................................... 12
Figure 2-6 Overview of the modules developed in this deliverable D3.3 on the global picture of the bIoTope project ........................................................................................................................................ 13
Figure 3-1 Database schema of the security module .................................................................................................. 15
Figure 3-2 Scenario for authenticating the user ‘Administrator’ and giving access to the permission management interface .................................................................................................................. 15
Figure 3-3 Scenario for authenticating a user and giving access to the O-DF resource ........................................... 16
Figure 3-4 Access management interface ................................................................................................................ 16
Figure 3-5 Scenario for giving access to a third-party application ........................................................................... 17
Figure 4-1 Seller scenario for publishing/selling his/her own IoT data/service thanks to the IoTBNB marketplace ....................................................................................................................................... 19
Figure 4-2 User authentication phase from the authentication-as-a-service component, i.e Auth0 ....................... 20
Figure 4-3 User Profile in the service catalogue ..................................................................................................... 20
Figure 4-4 O-MI server information from a seller perspective in the service catalogue ........................................ 21
Figure 4-5 Seller published data in the IoTBNB website .......................................................................................... 21
Figure 4-6 Buyer scenario for searching for/buying data/service in the IoTBNB website .................................. 21
Figure 4-7 Example of searching based on the keyword ‘moisture’ ....................................................................... 23
Figure 4-8 Example of the buyer cart .......................................................................................................................... 23
Figure 4-9 Payment step (not implemented yet) ......................................................................................................... 23
Figure 4-10 Buyer dashboard with the purchased data/service and related token ............................................... 24
Figure 4-11 Buyer scenario for searching for/buying data/service in the IoTBNB website with username anonymization ................................................................................................................... 24
Figure 4-12 Pseudonymization idea .......................................................................................................................... 25
Figure 4-13 k-anonymization idea .......................................................................................................................... 26
## Acronyms and Definitions

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACL</td>
<td>Access Control List</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>bIoTope</td>
<td>Building an IoT Open innovation Ecosystem for connected smart objects</td>
</tr>
<tr>
<td>HTML</td>
<td>HyperText Markup Language</td>
</tr>
<tr>
<td>HTTP</td>
<td>HyperText Transfer Protocol</td>
</tr>
<tr>
<td>IoT</td>
<td>Internet of Things</td>
</tr>
<tr>
<td>IoTbN</td>
<td>IoT service publication and Billing</td>
</tr>
<tr>
<td>IT</td>
<td>Information Technology</td>
</tr>
<tr>
<td>JS</td>
<td>JavaScript</td>
</tr>
<tr>
<td>JSON</td>
<td>JavaScript Object Notation</td>
</tr>
<tr>
<td>JVM</td>
<td>Java Virtual Machine</td>
</tr>
<tr>
<td>O-DF</td>
<td>Open Data Format</td>
</tr>
<tr>
<td>O-MI</td>
<td>Open Messaging Interface</td>
</tr>
<tr>
<td>PII</td>
<td>Personal Identifiable Information</td>
</tr>
<tr>
<td>REST</td>
<td>Representational State Transfer</td>
</tr>
<tr>
<td>RO</td>
<td>Read Only</td>
</tr>
<tr>
<td>RW</td>
<td>Read Write</td>
</tr>
<tr>
<td>SLA</td>
<td>Service Level Agreement</td>
</tr>
<tr>
<td>TCP/IP</td>
<td>Transmission Control Protocol / Internet Protocol (Protocol Stack)</td>
</tr>
<tr>
<td>UI</td>
<td>User Interface</td>
</tr>
<tr>
<td>UL</td>
<td>University of Luxembourg</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
<tr>
<td>WP</td>
<td>Work Package</td>
</tr>
<tr>
<td>XML</td>
<td>eXtensible Markup Language</td>
</tr>
</tbody>
</table>
Executive Summary

This deliverable is part of the scope of WP3 (Building a Secure, Open & Standardised Systems of Systems Platform for IoT), which provides the technological foundation of the bloTope Systems of Systems (SoS) Platform for information source publication and consumption in the IoT, based on the O-MI and O-DF standards. This includes new mechanisms to better manage ‘Identities’ and ‘Context-sensitive Security and Privacy’ (SaaS) of Connected Smart Objects and People to cope with the dynamic nature of the IoT.

Security, privacy and trust are still critical issues in today’s IoT. Even though there is no real consensus on how to implement security in the IoT, it is very important to define valid security, privacy and trust models that cope with IoT peculiarities to reach a full acceptance by the end-users. In that sense, the aim of this deliverable is to start with the technological building blocks that enable:

- end-users to access the O-MI/O-DF reference implementation in a secure way,
- third-party applications to easily access the O-DF resources published by the O-MI/O-DF reference implementation and to have a full authority to grant the access to their end-users
- a first level of end-user privacy management through the bloTope service catalogue (named IoTBNB)

As the deliverable D3.1 manage ‘Identities’ by using trust-based networks, this is the first deliverable dealing with security and privacy issues, which is intended to be more a technical report of the developed solutions than a state-of-the-art of the security solutions. The following deliverable, scheduled at month 29 (31st of May 2018), will focus on more complex scenarios in the future; i.e., when interacting with WP4 where ‘Contexts’ relates to human beings or physical objects will be taken into consideration (e.g., location, situation, level of trust or reputation of the surrounding entities and objects, etc...).
1. Introduction

Today, connected devices are increasingly present and interconnected in our everyday lives (e.g., weather stations, lights, baby monitors, cars, TVs, smart watches, smart shoes, bike stations, etc.). These devices are connected to the Internet through heterogeneous technologies, fostering the creation of innovative services in various domains including smart home, healthcare, energy, transportation, and so forth. However, security, privacy and trust are still critical issues in today’s IoT. For example, the security firm Kaspersky even talked about ‘Internet of Crappy Things’ [1], explaining how a hacker could wash his car free of charge or stalk someone by exploiting security breaches of a connected carwash or a connected fitness tracker. Another example is the exposure of the most private moments through baby monitors, as assessed by the security firm Rapid 7\(^1\) [2]. The crux of the issue is that today’s security, privacy and trust models are ‘Organization-centric’ rather than ‘Human centric’, leading to transparency and auditability issues, especially in Cloud-based environments. Indeed, even in the presence of Service Level Agreements (SLAs), current platform owners have very few incentives to develop platform privacy since the relationships with the consumer are mostly governed by unilateral contracts (i.e. provider defined). The white paper published by the security firm Wind River\(^2\) confirms this statement, in which the authors claim that the security is the foundational enabler of IoT but there is no real consensus on how to implement security in the IoT [3]. It is of the utmost importance to define soon, valid security, privacy and trust models that cope with IoT peculiarities to reach a full acceptance by the end-users [4]. In the literature, three major research tracks are presented [5]:

- ‘Whistleblower’ papers that point out security and privacy issues in the IoT (as for example the white papers or critiques above-cited),
- Technical papers that offer security provisions,
- Legal papers that develop framework about security-privacy regulations and laws in the IoT [6-7].

Common to all of the above research is the needs to answer the following questions: how to protect data exchanges between various information systems? how to manage authentication and access control in a world of billions of things? what about the privacy of end-users, and the security of the data generated by things? [8]. Vasilomanolakis & al. define a taxonomy of these IoT security challenges, based on five different groups (and their subcomponents) [9], namely:

- **Network Security:** Confidentiality, Integrity, Authenticity and Availability
- **Identity Management:** Authentication, Authorization, Accountability and Revocation
- **Privacy:** Data Privacy, Anonymity, Pseudonymity, Unlinkability
- **Trust:** Device trust, Entity trust and data trust
- **Resilience:** Robustness against attacks and Resilience against failures

Although most of the examples given above show security breaches at the device level, security considerations are orthogonal to the other research areas [10], and span all the levels of the IoT architectural framework as depicted in Figure 1-1, i.e. at the network level (local or Internet communications), gateway level as well as at the application level. Figure 1-1 depicts the generic IoT architectural framework that has been instantiated to the bioTope project. It shows the enabler components to achieve interoperability between vertical silos based on the O-MI/O-DF standards at the gateway level (i.e., considering O-MI nodes/servers at the edge network).

\(^1\) https://www.rapid7.com/
\(^2\) https://www.windriver.com/
As marked in Figure 1-1 this deliverable mainly focuses on security at the gateway level (i.e., at the bIoTope edge nodes that correspond to the O-MI servers/gateways), and will constitute our study perimeter. This deliverable is more a technical report of the developed solutions than a state-of-the-art of the security solutions. The objective is to lay the building block for the definition of more complex scenarios in the future; i.e., when interacting with WP4 where ‘Contexts’ related to human beings or physical objects will be taken into consideration (e.g., location, situation, level of trust or reputation of the surrounding entities and objects...).

Chapter 2 presents the current O-MI/O-DF reference implementation from a conceptual and technical viewpoint. A discussion about the security aspects is also provided to introduce the new security modules developed in this deliverable.

Chapter 3 presents the two independent security modules that are developed to enable end-users and/or third-party applications to access to the resources (O-DF payload) available through the O-MI node. First, authentication of the end-users is made available ‘as-a-service’ with the use of the external web service Auth0. Second, the authentication of the third-party applications is based on the JWT (Json Web Token) token technology.

Chapter 4 details how an IoT data/service provider can expose his/her own data/services to the service catalogue (developed in the bIoTope project) while benefiting from the security module and how an IoT data/service consumer can use the service that he/she has purchased from the service catalogue. In addition, a first level of privacy is implemented and relies on the end-users anonymization techniques.
2. O-MI reference implementation

2.1. Reminder on the current reference implementation

bIoTope consortium strongly believes that the adoption of any standard requires easy-to-use and easy-to-implement solutions to try and use it (i.e., avoiding having only written standards specifications). In this respect, the O-MI and O-DF standards were turned into an open reference implementation (available online: https://github.com/AaltoAsia/O-MI), the development activities of which are coordinated by Aalto University. Downloading this O-MI/O-DF reference implementation enables end-users, developers, businesses and other ecosystem stakeholders to download and deploy an O-MI node/server to better understand the standard specifications. They can jump straight into action by using the real request/response examples that cover all aspects of these standards.

The current reference implementation (i.e., also referred to as O-MI node in this document) consists of two main modules: the API endpoint (O-MI node server) and a UI (webclient) as depicted in Figure 2-1.

![Figure 2-1 O-MI reference implementation](image)

**API endpoint (O-MI node server):** The server implements all O-MI basic operations as described in Table 2-1. In addition, it maintains a database where the O-DF structure is stored. This structure is a hierarchical service description model with an ‘Objects’ element as its top element, which can contain any number of ‘Object’ sub-elements. ‘Object’ elements can have any number of properties, referred to as *Infotems*, as well as ‘Object’ sub-elements as shown in the Figure 2-2. O-MI/O-DF messages are self-contained, meaning that all the information necessary to enable the recipient to handle the message is contained within the message itself (e.g., the operation to be performed, the callback address, etc.). According to the standard specifications, an O-MI/O-DF request is sent through a HTTP POST request to a public URL (e.g., http://biotope.sntiotlab.lu:8080/ or https://otaniemi3d.cs.hut.fi/omi/node). The O-MI server acts as a “normal” REST endpoint for most of the O-MI operations, except for the subscription mechanism that allows either interval- or event-, or poll-based responses (cf. Table 2-1).
### Operation Description

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Write</strong></td>
<td>Used to send information updates to O-MI nodes.</td>
</tr>
<tr>
<td><strong>Read</strong></td>
<td>Used for immediate retrieval of information from an O-MI node.</td>
</tr>
</tbody>
</table>
| **Subscription** | **with callback address**: the subscribed data is sent to the callback address (basically a URL different from one that has sent the original request) at the requested interval. Two types of intervals are supported: **interval-based** and **event-based**;  
**without callback address**: data is memorized on the subscribed node as long as the subscription is valid. Historical data can be retrieved (i.e., polled) by issuing a new O-MI read request (by specifying the subscription ID). |
| **Cancel** | Used to cancel a subscription before it expires. |

#### Table 2-1 Main Messaging Interfaces specified in the O-MI standards

**User Interface (webclient):** A graphical interface is accessible through the O-MI node server website (which is nothing than the API endpoint, e.g., [http://biotope.sntiotlab.lu:8080/](http://biotope.sntiotlab.lu:8080/) or [https://otaniemi3d.cs.hut.fi/omi/node](https://otaniemi3d.cs.hut.fi/omi/node)) as depicted in Figure 2-3.

**Figure 2-3** Webpage of the O-MI webclient (here for the node’s URL [http://biotope.sntiotlab.lu:8080/](http://biotope.sntiotlab.lu:8080/))

Any IoT data/service consumer has three different URLs, as highlighted in Figure 2-3:

- The first URL enables to better understand how to discover (and navigate through) the O-DF tree, where messages are based on HTTP GET requests. For example, in Figure 2-4, the data/service consumer has discovered that in the Object ‘SnT’, there is an Object named ‘IoTlab’, in which an Object named ‘Netatmo’ is defined, which in turn contains 8 infoltems (i.e., Object attributes), and so on;
• The second URL is the core of the UI, helping data/service consumers to play with the O-MI/O-DF standard specifications (i.e., sending/receiving and most importantly visualizing O-MI/O-DF requests/responses);
• The third URL just provides an example of syntax of the O-MI/O-DF request & response.

From a deliverable perspective, we will rely mainly on the second URL since it offers a user-friendly UI that will help the reviewer to better follow the deliverable’s flow. This UI (given in Figure 2-5) aims to guide step-by-step the data/service consumer when creating his own O-MI/O-DF request. All the user interaction occurs in the left panel. To visualize the set of IoT data/services that are exposed/published by the O-MI node server, the webclient sends a request to the API endpoint (see ‘O-DF Structure’ panel in Figure 2-5). The end-user can therefore navigate through the O-DF tree and select as he/she sees fit the desired data/service. Then, all operations supported by the O-MI standard (see ‘O-MI Request’ panel in Figure 2-5) can be solicited by specifying the needed ‘parameters’. In the same way, the corresponding XML request is generated and sent to the API end-point (the example given in Figure 2-5 shows the value for the requested Infoltem ‘TemperatureIndoor’).

**Agent (wrapper):** As depicted in Figure 1-1, the gateway level and particularly the O-MI node server can implement one or more agent(s)/wrapper(s) to plug any existing platform/system/silo to the O-MI node and populate the real value(s)/service(s). (The platform’s API is used to access the data source and wrap it into an understandable and standardized format, namely O-DF.) From an implementation perspective, two types of agents are defined: internal and external. Either they run on the same Java Virtual Machine as the O-MI server (internal agent) or they run as independent processes (external agent). Let us note that the external agent can be located on the same device or a different one. External agents can be implemented in any programming language that supports the TCP/IP library.
Figure 2-5 O-MI webclient UI after clicking 'Read All' (with the published O-DF tree), and selecting the InfoItem TemperatureIndoor in the Object(s) SnT, IoTab and Netatmo
2.2. Discussion about the security in the reference implementation

From a security standpoint, the O-MI/O-DF reference implementation has a first level of access control, giving the possibility to the local administrator of the node to define what peer systems (IP-based) can access the O-MI node server in write access (all peer systems can, by default, access the whole O-DF tree in read only).

A first step to secure the O-MI node consists in implementing an access control and authentication mechanism for controlling what third party end-users or systems can access the O-MI node server. This step is detailed in the section 3-1. Then, an additional module has been developed to enable such third-party persons/applications to access the O-MI node through their application. A first token-based access control solution is proposed and described in section 3-2. From a service catalog viewpoint – which is where third party developers will search for valuable IoT data/service and get the necessary information to access the IoT data/service from a remote O-MI node server (if the necessary conditions are in place), as will be presented in greater detail in section 4 – the frontend and backend systems should integrate and handle this new token-based management solution. Finally, as a complementary module, a first privacy management strategy is developed based on anonymization techniques, as will be explained in the section 4.2.

Figure 2-6 Overview of the modules developed in this deliverable D3.3 on the global picture of the bioTope project
3. Security modules for the O-MI node

The development and implementation stages follow the microservice architectural style [11], which is a ‘fine-grained SOA (Service-Oriented Architecture)’ as defined by Adrian Cockcroft (a cloud architect at Netflix) in a conference talk. To put it simply, a microservice architecture is a distributed application where all modules are minimal independent processes interacting via messages (i.e. microservices) [12]. This is a way of structuring many related applications to enable them to work together rather than creating a unique application in which modules - also called monoliths - are dependent on the expected application to which they belong.

Given the above, two independent security modules are developed to enable end-users and/or third-party applications to access the resources (O-DF payload) available through the O-MI node. The core idea here is to avoid making basic security mistakes that most of today’s IoT device manufacturers usually make. For example, communications should be encrypted (referring to a cleartext API), local accounts should not use easily guessed passwords (backdoor accounts), and so on.

3.1. Access control and authentication mechanism for the end-users of the O-MI webclient

The objective is to provide end-users with a secure access to the O-MI webclient. To this end, the following requirements are identified:

- **Access control of the resources**: only users who have the access rights can perform the corresponding O-MI/O-DF request actions over the service tree;
- **Group-based rules**: all end-users must belong to a group, and then access rules are specified for each group (never for single users). Every end-user can be a member of one or more groups;
- **Permissions according to the O-MI verbs**: Essentially this requirement can be translated into the well-known distinction between RO and RW access rights:
- **Recursive permission mechanism**: can be inherited from the parent’s Object (same as in modern file systems) as well as overridden for particular children;
- **User & rule management interface**: the system administrator must be able to control access right policies through a centralized interface.

To achieve the above features, the security module developed in bIoTope consists of two sub-modules, namely a database and an external authentication service, which must provide services for:

- **Permission management (through a UI)** to enable the administrator to define the access rights (according to the registered users),
- **An access control service** to verify whether a given end-user has the right permissions on the requested information,
- **A separate database** to store end-users/groups and their related access right settings;
- **Authentication** to be made available ‘as-a-service’ with the use of the external web service Auth0 (http://auth0.com).

Before explaining the interaction between these three components, let us present the database schema used to store the access rules. As depicted in Figure 3-1, the database contains three main tables: **user**, **group** and **rule**. A user is characterized by his username and email only, and belongs to a specific group (users and groups have relationship achieved through a helper table named USER_GROUP_RELATION). Access rules are associated to a group, which are defined according to the:
- Data object: path of the O-DF XML structure (e.g. ‘Objects/SmartHouse/FrontDoor’) named in this table hierarchy_id or hid;
- Allowed operation: cf. Write_Permissions column, that is a Boolean flag: 0=read only or 1=read & write;
- Object type: i.e. Object or InfoItem in the O-DF standard on which must be applied the rule. The Administrator table is a helper table that maintains a list of users with admin rights and is filled manually by the server administrators.

**Figure 3-1 Database schema of the security module**

To understand the interaction between these components, let us consider two scenarios depicted in Figures 3-2 and 3-3 depending on whether the user is or not the administrator. In both cases, the user accesses the O-MI webclient (UI) through a web browser and needs to log in to be able to visualize and access the part of the O-DF tree to which it is allowed to access to. The authentication (denoted by circle 1 in Figure 3-2) is done from the Auth0 service (enabling the user to log in with Auth0 credentials) or through social network provider credentials (e.g., Google, Facebook). Once the user is logged in, the module registers the user in the database (username and email being obtained from the Auth0 response), thereby creating a user session accordingly (i.e. a session cookie).

**Figure 3-2 Scenario for authenticating the user ‘Administrator’ and giving access to the permission management interface**
When the user tries to access a specific resource (permission management or O-DF information), the O-MI node interacts with the access control service (cf. circle denoted by 2 in Figure 3-1 and 3-3), asking whether ‘user Y can access resource X considering the type of request received’. The service’s answer (‘Yes or No’) depends on the rules that the administrator has initially specified. At a more technical level, an interface of the O-MI node called AuthAPIService is responsible to check whether the request contains a session cookie and, if so, sends the HTTP POST request to the access control service with the following parameters: (list of objects, user information, user session). If the answer is ‘Yes’, then the requested O-DF resource is displayed to the user (cf. circle denoted by 3 in Figure 3-3), otherwise an ‘Unauthorized’ response code is returned (return code 401 in the O-DF payload).

![Figure 3-3 Scenario for authenticating a user and giving access to the O-DF resource](image)

The above-mentioned interface looks like the O-MI webclient, except that it has been customized to some extent, as shown on Figure 3-4. Through this UI, the administrator can manage users, groups (e.g., creating, modifying or deleting groups) and associated access rights. The ‘Save’ button results in the storage of the specified users/groups/rules into the database thanks to the permission management service (cf. circle denoted by 3 in Figure 3-2).

![Figure 3-4 Access management interface.](image)
When an OMI node’s administrator deploys this security module, the HTTPS protocol is used for encryption and integrity check of all communications between the web browser and the O-MI node server. From a practical viewpoint, the OMI node’s administrator can configure the ports related to the HTTP port of the AuthAPIService (in charge of sending the permission request to the access control service) and the HTTPS port (e.g. port 8089 in the Figure 3-4) to give access to the O-MI webclient and to the permission management interface. Let us note that this module does not substitute the security rules defined at the network/communication level (firewall, Access-list in the network infrastructure, etc.).

3.2. Secure O-MI/O-DF information exchange for third party applications

The objective is to provide third-party consumers with a secure access to the O-MI node server. To this end, the two following requirements are identified:

- **Access control of the resources**: only third-party application owning a token can access O-DF-related resources according to the token-related access rights,
- **Full authority of the third-party applications**: once the third-party application is allowed to use the token management service, it can give access to its end-users through a specific token.

Given the above requirements, the security module developed consists of two services, namely:

- A **token delivery service** provision to enable third-party application to get a token (for a specific end-user),
- A **token verification service** provision to verify if the consumer has the access rights on the requested information.

Let us consider the scenario depicted in Figure 3-5, in which a third-party application wants to access some O-DF resources. It first needs to ask for a token by sending an HTTP POST request (in a JSON format) with the parameters specified in Table 3-1 (cf. circle denoted by 1). The token delivery service returns a **JSON Web Token (JWT)**, which is based on an open standard (RFC 7519) defining a compact and self-contained way for securely transmitting information as a JSON objects between parties. This token consists of three distinct parts, namely the (i) header, (i) payload and (iii) signature, as presented in greater detail below.

![](image)

**Figure 3-5 Scenario for giving access to a third-party application**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>clientId</strong></td>
<td>The identifier of the third-party application giving by the O-MI node</td>
</tr>
<tr>
<td><strong>clientSecret</strong></td>
<td>The secret key of the third-party application giving by the O-MI node</td>
</tr>
<tr>
<td><strong>hid</strong></td>
<td>The path in the O-DF XML structure, e.g. ‘Objects/SmartHouse/FrontDoor’</td>
</tr>
<tr>
<td><strong>username</strong></td>
<td>The username of the O-DF information requester</td>
</tr>
<tr>
<td><strong>validity</strong></td>
<td>The time (in second) after which the token will expire</td>
</tr>
</tbody>
</table>

Table 3-1 Parameters for asking the token
**Header:** it specifies the message authentication algorithm, e.g. `{alg: ‘HS256’}` means that the message is authenticated and integrity-protected using the HMAC SHA-256’s algorithm. The corresponding JSON is (Base64Url) encoded to form the first part of the JWT as follows: eyJhbGciOiJIUzI1NiJ9.

**Payload:** This is the second part of the token containing the claims that are statements about the exchange. For instance, the implemented module uses this following payload:

```json
{
  ‘hid’: ‘Objects/SmartHouse/FrontDoor’,
  ‘iss’: ‘ominode’,
  ‘exp’: 1484905924,
  ‘iat’: 1484895924,
  ‘username’: ‘test’
}
```

where metadata ‘path’ (or also named hid in previous sections) represents the path that can be accessed to (obtained in the request), ‘iss’ the name of the issuer (here the OMI node’s name), ‘exp’ the date by which the token will expire (computed based on the issue date et validity requested), ‘iat’ the date by which the token is issued, and ‘username’ (obtained in the request) for statistical purposes. This JSON is also encoded to form the second part of the JWT, resulting in:

eyJoWQOiOiJPYmplY3RzL1ntYXJ0SG91c2UvRnJvbnREb29yLiwiXNlzoiib2lpm9kZSIslmV4eCl6MTQ4NDkwNTkyNCwiWF0ljoxNDg0ODk1OTI0LCJ1e2VybmFiZSI6InRlc3QiOQ==

**Signature** This is created by taking the encoded header, payload, secret and algorithm specified in the header. It makes it possible to verify both that the JWT’s sender really is the entity it purports to be and that the message was not changed/corrupted along the way. The returned JWT would therefore become:

eyJhbGciOiJIUzI1NiJ9.eyJoaWQiOiJPYmplY3RzL1ntYXJ0SG91c2UvRnJvbnREb29yLiwiXNlzoiib2lpm9kZSIslmV4eCl6MTQ4NDkwNTkyNCwiWF0ljoxNDg0ODk1OTI0LCJ1e2VybmFiZSI6InRlc3QiOQ==

The third-party application sends the O-MI/O-DF request to the API endpoint (via an HTTP POST request) including the JWT as the Authorization header (cf. circle denoted by 2 in Figure 3-5). The token verification service checks whether the JWT is a valid one and, if so, allows the third-party access to the protected resources (cf. circle denoted by 3).
4. Security & Privacy with the service catalogue IoTBnB

Let us recall that the main objective of the service catalogue (IoTBnB) is to enable person and/or system to [13]:

- **IoT data/service producers** *(consumers of goods, citizens, municipalities, businesses or other legal entities)* for having the possibility for joining the bioTope ecosystem community and describing – based on O-MI/O-DF – what personal IoT data/services from their own digital environment are available, and how to query/call them, while being able to control how and by whom those data/services can be accessed, processed, re-used, re-published, etc.;

- **IoT data/service consumers** *(businesses are of course the first parties concerned like integrators device manufacturers, software companies...)* with the possibility and motivation for joining the marketplace community, searching for, and consuming valuable IoT data streams and/or services with the objective to re-use/compose them – under certain conditions/IPRs and potential financial returns – into new and sustainable developments that fulfill untapped needs and applications.

It is important to understand that IoTBnB does not act as a Cloud data storage center where IoT data generated by smart connected objects is stored, but rather as a service registry web space (or marketplace) where the ‘descriptions’ of IoT data/services are stored, indexed and searchable. Then, only once the consumer wants, and is allowed to access it, IoTBnB plays the role of third-party application, putting the publisher and consumer in relation with each other so that data/services can be performed in a peer-to-peer manner (without transiting through IoTBnB). Having said that, the objective of this section is to detail:

- How an IoT data/service provider can expose his/her own data/services to the service catalogue while benefiting from the security module previously introduced;

- How an IoT data/service consumer can use the service that he/she has purchased from the service catalogue;

- How the service catalog can anonymize an end-user or an application as first level of privacy.

4.1. Security management with IoTBnB

From a security perspective, IoTBnB should be able to obtain the JWTs from the O-MI node that holds the IoT data/service to provide it to the consumer of that data/services. To do so, the website needs to have access to the seller’s API to get tokens on-behalf of the buyers as a third-party application.

![Figure 4-1 Seller scenario for publishing/selling his/her own IoT data/service thanks to the IoTBnB marketplace](image-url)
Figure 4-1 describes the scenario from the perspective of an IoT data/service publisher, who wants to make searchable his/her data/service in the service catalogue. The assumption in this scenario is that this publisher has implemented the security for third-party application (as described in section 3.2.) and signed-up or logged in to the service catalogue (cf. circle denoted by 1 in Figure 4-1). As shown in Figure 4-2, IoTbnB requests the authentication-as-a-service component (i.e. Auth0 service) to obtain the publisher’s profile (cf. circles denoted by 2 and 3). In the current version of IoTbnB, only the username and email address are displayed and stored (cf., Profile page in Figure 4-3). Following this step, the data/service publisher fills out the form about the needed information related to his O-MI server and the security module (cf. circle number 4). Figure 4-4 presents this form in which the O-MI node’s URL, name and location are required, as well as the URL of the security module, client ID and secret. At this stage, IoTbnB can request a token to access the whole O-DF structure (cf. circles denoted by 5 and 6) to be able to harvest and index the set of IoT data/services exposed/described by the publisher’s O-MI node server. Let us note that this token is not needed if the seller does not implement the security module. Then, the publisher can select the O-DF data/services that he/she wants to publish/expose through the public service catalogue (cf. circle denoted by 8), along with a specific price for accessing it, as emphasized through Figure 4-5 (in the project, a framework supporting crypto-currency technologies will be tested, and particularly considering the Bitcoin technology).

3 The indexing is carried out based on OpenDataSoft Software-as-a-Service platform, which can be accessed as long as the token provided by IoTbnB is valid (cf. circles denoted by 7’ and 7’).
Figure 4-4 O-MI server information from a seller perspective in the service catalogue

Figure 4-5 Seller published data in the IoTBNB website

Now, let us consider the buyer viewpoint.

Figure 4-6 Buyer scenario for searching for/buying data/service in the IoTBNB website
Figure 4-6 describes how an IoT data/service consumer can search for a data/service through the IoTBnB service catalog (API currently being specified), buy it and consume it. Similarly to the publisher, the consumer of IoT data/services can sign-up on the service catalogue (cf. circle denoted by 1) via the authentication-as-a-service component (cf. circles denoted by 2 and 3). The first page of IoTBnB displays all the indexed data/services (developed based on the Opendatasoft widget/component suite), where such data/services can be searched in a multimodal manner, e.g.:

- **Spatial/Temporal search**: one may want to search for services within a geographical area;
- **Keyword search**: one may want to search for services falling in a specific sector (e.g., mobility, healthcare, building, etc.). Such a feature requires an efficient and scalable way to index IoT services, which is partly dependent on what semantic web vocabularies have been used in the O-DF service description model/tree.
- **Reputation search**: one may want to search for a service ensuring a certain level of quality, which comprises various dimensions such as (i) *data quality*: quality of sensor data streams or more advanced services like how accurate a failure prediction algorithm is, (ii) *service owner reputation*: third party developers being able to leave a review about whether a given data stream works well, or the publisher’s reactivity when asked questions, etc. Such a reputation will be computed thanks to the service quality assessment framework developed in T3.C;
- **Contractual term or Technology search**: one may want to search only for IoT data/service producers that make available data/service for free, or are compliant with one or more crypto currencies, or data/services that are compliant with specific IPR policies (i.e., license type, etc.).

Figure 4-7 shows an example where the end-users have searched for the keyword ‘moisture’, resulting in two available data/services. Let us consider that the consumer wishes to buy the first one: it is added to the cart, the accessibility duration is specified (cf. Figure 4-8), the payment transaction is performed in an ad-hoc manner between the data/service consumer and publisher, and monitored by IoTBnB (cf. Figure 4-9). If the transaction is successful, IoTBnB gets a token from the data/service publisher (based on the information entered by the publisher at the registration phase), which is then given/displayed in the consumer’s dashboard (where he/she can see/use the set of tokens related to all the purchased data/services, as shown in Figure 4-10). Then, the IoT data/service consumer can start exchanging/requesting data/services using the O-MI operations with remote publisher’s O-MI node server (the token being included in the HTTP header request). This paves the way to a more secure and trustable data exchange following a ‘human centric’ security model.
Figure 4-7 Example of searching based on the keyword ‘moisture’

Figure 4-8 Example of the buyer cart

Figure 4-9 Payment step (not implemented yet)
4.2. Privacy management strategies with IoTBnB

From the service catalogue perspective, one of the main concerns is to ensure the privacy of its users, and especially in the exchanges between seller and buyer. Indeed, as seen in the previous sections, when a buyer wants to consume purchased data, he/she uses a token in which a username (e.g. an email address) is included and requested by the O-MI node. This constitutes a personal identifiable information (PII), which can be considered sensitive. The service catalogue needs therefore to ensure that the seller does not obtain the buyer’s PII when the buyer gains access to ODF resources. There are several techniques that attempt to protect privacy of an individual such as anonymization, pseudonymization, encryption and so on. The first techniques are the approaches mainly used (in particular in the healthcare sector).

Anonymity can be defined as ‘the state of being not identifiable within a set of subjects, the anonymity set’ [14]. The objective of the anonymization is therefore to remove of as much information as needed to conceal the user’s identity. For example, PII such as name, zip code, date of birth, address and so on would be removed when there is no need for them [15]. Pseudonymization can be defined as the use of false names (or pseudonyms) in the process of masking the user’s identity so that information relating to this user can be used without knowing to whom the information relates [16].

Figure 4-10 Buyer dashboard with the purchased data/service and related token

Figure 4-11 Buyer scenario for searching for/buying data/service in the IoTBnB website with username anonymization
As a first level of user privacy, the objective is to implement pseudonymization and anonymization techniques in the service catalogue. These can be implemented either as internal or external in the IoTBNB service catalogue. Based on the scenario depicted in Figure 4-6, Figure 4-7 shows how an anonymization service can be easily integrated. Let us note that this service will be activated only when the buyer wants to be anonymous. (This option will be made available through his/her IoTBNB dashboard later.) In that case, after having paid for the data, the service catalogue requests an anonymised username to the anonymization service (cf. circle denoted by a). Based on the input parameter username (i.e. the buyer’s email address), the service can return an anonymized username (cf. circle denoted by b).

A first strategy, named **pseudonymization**, consists in binding the username to a pseudonym. As depicted in Figure 4-12, every username in the IoTBNB domain is associated to a pseudonym in the anonymization service domain. For example, the user with the email address camille@free.fr will be mapped to the pseudonym buyer-2. This translation is then stored in a table to be able to return the same pseudonym for the same user. The seller can therefore compute usage statistics – if he/she wants. In that purpose, the seller can track the data access by reading a file – named anonymization – in the logs/ directory of the O-MI node. This file consists of two pieces of information: (i) the user identifier (anonymized or not) and (ii) the path of the consumed O-DF resource.

![Figure 4-12 Pseudonymization idea](image)

A second strategy, named **k-anonymization**, consists in mapping the username to a ‘quasi-identifier’ that refers to a group where there are at least k records with the same occurrence. According to the number of records in the table, either the suppression method or the generalization method will be applied. The suppression method associates the username to a specific character such as, for example, an asterisk (‘*’). The generalization method consists in mapping the username to a category name to k records. For example, the anonymization service could rely on the institution-based categories; this means that the chosen category will be the name of the institution given in the email address (i.e. the word after the character ‘@’ and before ‘.’). In this case, Figure 4-13 shows the both methods where:

- the usernames jeremy@uni.lu and kary@aalto.fi are both replaced to ‘*’ following the suppression method, since there are no records with the same institution (either uni or aalto);
- the usernames camille@free.fr and titi@free.fr are translated to free following the generalization method.

Let us note that only 2-anonymity is ensured since there are only 2 records with the same category. The parameter k will be larger when the number of the IoTBNB users increases. However, this parameter could also be chosen by the user in his/her dashboard.
As of the time of writing, the implementation choices are not set in stone but the both approaches explained have been implemented as a feasibility study. However the privacy is still a challenge in the bioTope eco-system, and more broadly in the IoT. Consequently, even though this gives a first overview of the privacy strategies that could be implemented in the next release of the IoTbB service catalogue, a further study (and a more complete state-of-the-art) would be necessary to make the right implementation choice for future developments.
5. Conclusion

This deliverable provides insight into the two first building blocks/modules to secure information exchanges at the gateway level for the end-users and the third-party applications. The first module relies on the external authentication service Auth0, which enables end-users to authenticate themselves in the O-MI/O-DF reference implementation webclient using their Auth0 or social network provider credentials. The second module enables third-party applications to access O-DF resources through the REST interface by using a JWT. This first release gives full authority to the third-party applications to manage the access right. However, a more fine-grained version could benefit from the two modules to verify whether a particular end-user (i.e. the third-party application based on a username which will be included in the token) is allowed to access a particular resource (included in the token as well). This would need that the modules communicate with each other. In addition, access control policies could be more advanced by considering one or more ‘Contexts’ related to a human being or a physical object (e.g. location, situation, level of trust or reputation of surrounding entities...) [17]. In the current version of the O-MI/O-DF reference implementation, a very basic level of context security based on the IP address for the specific write operation is implemented.

This deliverable also highlights how the security module is used with the bloTope service catalogue (IoTBnB). The third-party application IoTbnB can get a token from the data/service publisher (i.e. the seller) on behalf of the consumer (i.e. the buyer). The consumer can therefore see/use the set of tokens related to all the purchased data/services in his/her dashboard. In addition, the service catalogue offer a first level of privacy to the consumer, which can be anonymous – if he/she wants and according to his/her desired level – from the publisher viewpoint. In future work, the state of the art of the privacy models/strategies (e.g., k-anonymisation) at research level as well as at practical level will be further studied to give more privacy options to the users. This will therefore lead to a more complete ‘Human centric’ security and privacy model.
6. References


