Project Acronym:	**VICINITY**
Project Full Title:	**Open virtual neighbourhood network to connect intelligent buildings and smart objects**
Grant Agreement:	**688467**
Project Duration:	**48 months (01/01/2016 - 31/12/2019)**

Deliverable D3.3

Open Interoperability Gateway API, first version

Release Note 1.0

Work Package:	**WP3 – VICINITY Server Implementation**
Task(s):	**T3.1 – VICINITY core components implementation**
Lead Beneficiary:	**BVR**
Due Date:	**30 June 2017 (M18)**
Submission Date:	**6 July 2017 (M19)**
Deliverable Status:	**Final**
Deliverable Type[1]:	**R and DEM**
Dissemination Level[2]:	**PU**
File Name:	**VICINITY_D3.3_Open_Interoperability_Gateway_API_Release_note_v1.0.pdf**

VICINITY Consortium

No	Beneficiary	Short Name	Country
1.	TU Kaiserslautern (Coordinator)	UNIKL	Germany
2.	ATOS SPAIN SA	ATOS	Spain
3.	Centre for Research and Technology Hellas	CERTH	Greece
4.	Aalborg University	AAU	Denmark
5.	GORENJE GOSPODINJSKI APARATI D.D.	GRN	Slovenia
6.	Hellenic Telecommunications Organization S.A.	OTE	Greece
7.	bAvenir s.r.o.	BVR	Slovakia
8.	Climate Associates Ltd	CAL	United Kingdom
9.	InterSoft A.S.	IS	Slovakia
10.	Universidad Politécnica de Madrid	UPM	Spain
11.	Gnomon Informatics S.A.	GNOMON	Greece
12.	Tiny Mesh AS	TINYM	Norway
13.	HAFENSTROM AS	ITS	Norway
14.	Enercoutim – Associação Empresarial de Energia Solar de Alcoutim	ENERC	Portugal
15.	Municipality of Pylaia-Hortiatis	MPH	Greece

[1] ***Deliverable Type:***

R:	Document, report (excluding the periodic and final reports)
DEM:	Demonstrator, pilot, prototype, plan designs
DEC:	Websites, patents filing, press & media actions, videos, etc.
OTHER:	Software, technical diagram, etc.

This project has received funding from the European Union's Horizon 2020 Research and innovation programme under Grant Agreement n°688467

[2] ***Dissemination level:***

PU:	Public, fully open, e.g. web
CO:	Confidential, restricted under conditions set out in Model Grant Agreement
CI:	Classified, information as referred to in Commission Decision 2001/844/EC.

Authors List

Leading Author (Editor)				
Surname		First Name	Beneficiary	Contact email
Oravec		Viktor	BVR	viktor.oravec@bavenir.eu
Co-authors (in alphabetic order)				
No	**Surname**	**First Name**	**Beneficiary**	**Contact email**
1.	Vanya	Stefan	BVR	stefan.vanya@bavenir.eu
2.	Martin	Horniak	BVR	martin.horniak@bavenir.eu
3.	Hana	Hozzankova	BVR	hana.hozzankova@bavenir.eu
4.	Kostelnik	Peter	IS	Peter.kostelnik@bavenir.eu
5.	Jorge	Almela	BVR	jorge.almela@bavenir.eu
6.	Serena	Fernando	UPM	fserena@fi.upm.es
7.	Mynzhasova	Aida	UNIKL	limit6715@gmail.com
8.	Theologou	Natalia	CERTH	nataliath@iti.gr
9.	Tsiggenopoulos	Ilias	CERTH	itsiggen@iti.gr
10.	Savaghebi	Mehdi	AAU	mes@et.aau.dk
11.	Bračko	Mihael	GRN	mihael.bracko@gorenje.com
12.	Frengstad	Olav	TINYM	olav@tiny-mesh.com
13.	Hovsto	Asbjorn	HITS	hovsto@online.no

Reviewers List

List of Reviewers (in alphabetic order)				
No	Surname	First Name	Beneficiary	Contact email
1.	Vinkovič	Sašo	GORENJE	saso.vinkovic@gorenje.com
2.	Garcia-Castro	Raul	UPM	rgarcia@fi.upm.es
3.	Chochliouros	Ioannis	OTE	ichochliouros@oteresearch.gr

Revision Control

Version	Date	Status	Modifications made by
0.1	30 May 2017 (M17)	Initial Draft	Viktor Oravec (BVR)
0.2	5 June 2017 (M18)	Draft	Peter Kostelnik (IS)
0.3	7 June 2017 (M18)	Quality Check	Viktor Oravec (BVR)
0.3.1	13 June 2017 (M18)	Quality Check	Martin Horniak (BVR)
0.4	27 June 2017 (M18)	Final Draft reviewed	Viktor Oravec (BVR), Ioannis Chochliouros (OTE), Raul Garcia-Castro (UPM), Sašo Vinkovič (GRJ), Peter Kostelnik (IS), Martin Horniak (BVR)
1.0	6.7.2017 (M19)	Submission to the EC	Viktor Oravec (BVR), Carna Radojicic (UNIKL)

Executive Summary

The present document is a deliverable of the VICINITY project, funded by the European Commission's Directorate-General for Research and Innovation (DG RTD), under its Horizon 2020 Research and Innovation Programme (H2020).

The D3.3 – VICINITY Open Interoperability Gateway API first version - is the first release (i.e., the Alpha release) of all related VICINITY Releases. The goal of the D3.3 – VICINITY Open Interoperability Gateway API first version is to provide critical functionality of the VICINITY Gateway API to proceed in early concept validation for integrated infrastructure in the VICINITY Pilot sites and the VICINITY value-added services.

Based on the VICINITY Architectural design, expected VICINITY Open Interoperability Gateway API functionality is the following:

- *Registry service* – set of services that enables to register, access and that provides IoT objects descriptions;
- *Consumption service* – set of services to interact with IoT objects in the following ways:
    - Get and set IoT objects properties;
    - Performing the actions on the IoT objects;
    - Receiving the events of the IoT objects;
- *Discovery and query service* – set of services to perform discovery in the VICINITY virtual neighbourhood and query for IoT objects' properties, based on IoT objects semantic description.

According to "Interface (integration) view and Process view" in D1.6 VICINITY Architectural design", *Registry services* and *Discovery and query services* dependents on the VICINITY Neighbourhood Manager and Semantic discovery & dynamic configuration agent platform, including Gateway API Service components. These components are in early stage of implementation, thus these features will be provided in upcoming release.

Thus, the first version of the VICINITY Gateway API is focused on *Consumption services* to exchange properties and actions between integrated IoT infrastructures and value-added services. This will enable us to practically proceed to the proof of a concept on integration of value-added services and IoT infrastructures.

Current implementation of VICINITY Gateway API (including examples of adapters and integration documentation) can be downloaded from the VICINITY webpage (http://vicinity2020.eu/vicinity/content/implementation).

Remaining functionalities will be published in the next official so called as "Beta" release of the VICINITY Core components, planned in M30 (in line with DoA) of the VICINITY project.

Table of Contents

List of Tables

List of Figures

List of Definitions & Abbreviations

Abbreviation	Definition
API	Application Programming Interface
CRUD	Create, Read, Update and Delete
DG RTD	Directorate-General for Research & Innovation
DoA	Description of Action
EC	European Commission
EU	European Union
GIT	Version control system for tracking changes of computer files
GitLab	Software platform managing GIT repositories
H2020	Horizon 2020
HTTP	HyperText Transfer Protocol
ID, id	Identifier
IoT	Internet of Things
IP	Internet Protocol
JDK	Java Development Kit
JSON	JavaScript Object Notation
P2P	Peer-to-Peer
RC1	Release Candidate 1
SLACK	Real-time messaging system for software developers.
URL	Uniform Resource Locator
VIC	VICINITY
WLAN	Wireless Local Area Network
XML	Extensible Mark-up Language
XMPP	Extensible Messaging and Presence Protocol

1. Introduction

This document includes the Release Note of D3.3 deliverable. The whole deliverable is divided in the following sections:

- *Section 2:* Overview of VICINITY architecture design, list of releases of core VICINITY components and description of VICINITY development environment, to understand how the implementation of the VICINITY core components are managed.
- *Section 3:* Short overview of the D3.3 release defines the scope of this release;
- *Section 4:* List of issues implemented in the current release summarizes the break-down of the release into the task;
- *Section 5:* List of bugs fixed in the current release;
- *Section 6:* Release application instruction describes how the VICINITY Gateway API needs to be installed;
- *Section 7:* Documentation notes (such as updates of the architecture design or functional specification amendments);
- *Section 8:* List of the source code changes in relevant repositories;
- *Section 9:* Conclusions.

2. Overview of architecture

VICINITY architecture – as described in the respective deliverable D1.6 - defines a decentralised interoperability between integrated IoT infrastructures and value-added services (called as the virtual neighbourhood) through a peer-to-peer (P2P) network of VICINITY Nodes. The VICINITY Nodes integrate IoT infrastructures and value-added services into VICINITY and provide access points to access IoT objects and services through the VICINITY peer-to-peer network. The VICINITY Cloud provides specific features to: setup connection of integrated IoT infrastructures and value-added services to VICINITY; register new devices; deploy new value-added services, and; set-up interoperability between IoT objects.

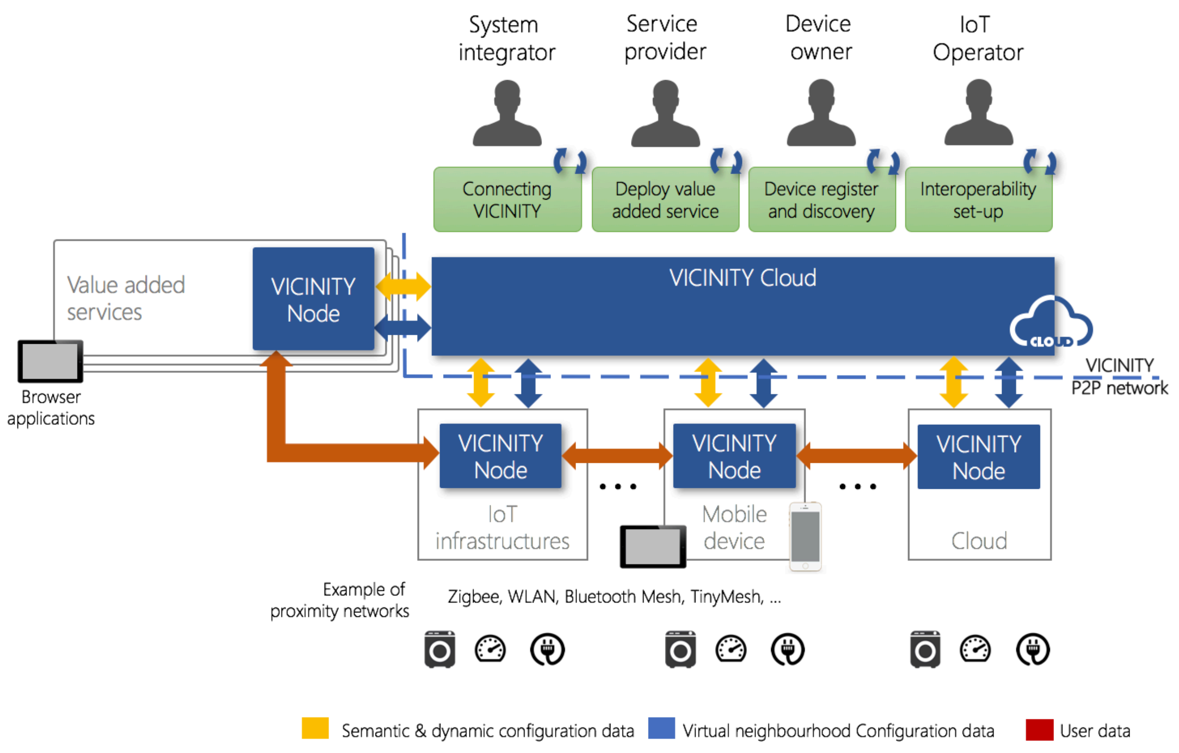

Figure 1: VICINITY Overall Architecture

2.1. VICINITY Cloud

The VICINITY Cloud constitutes of the following essential components:

- *VICINITY Neighbourhood Manager* – it is a principal VICINITY user interface;
- *Semantic discovery & dynamic configuration agent platform* – it is a semantic repository for the discovery of IoT objects;
- *Gateway API service*, managed as part of Open Interoperability Gateway API releases – It is a supporting service for discovery and query services;
- *VICINITY Communication Server*, managed as part of High-available VICINITY server deployment releases – This controls the communication between integrated IoT communication.

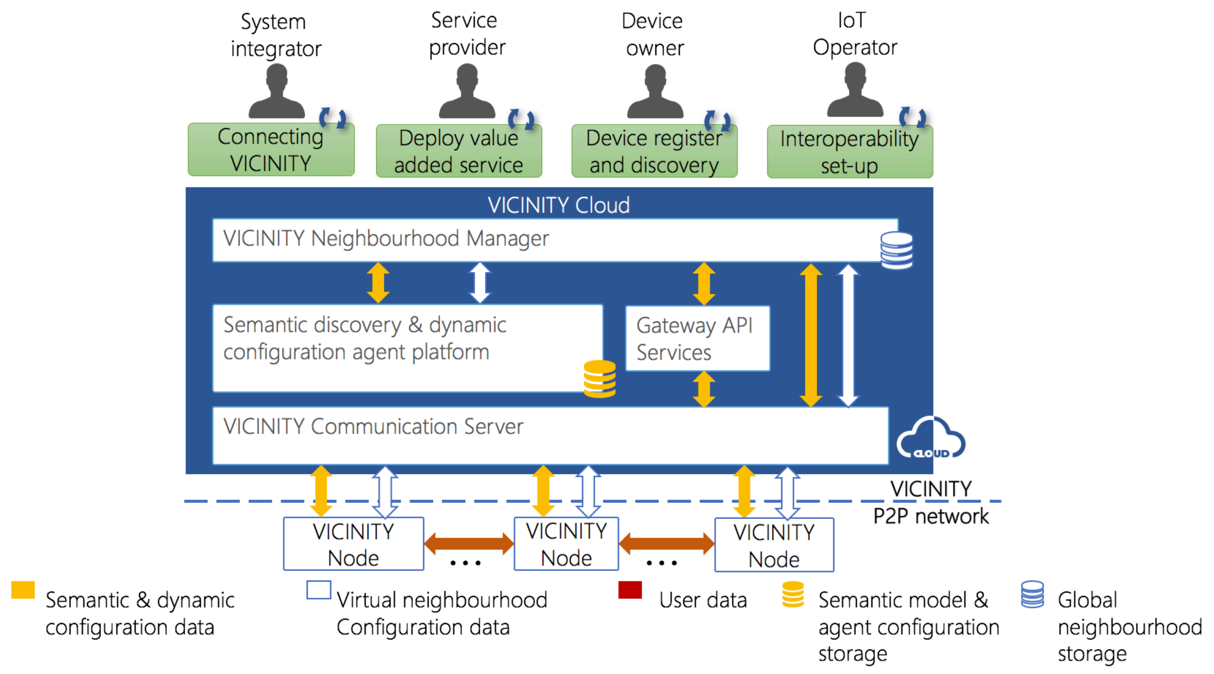
System integrator
Service provider
Device owner
IoT Operator
Connecting VICINITY
Deploy value added service
Device register and discovery
Interoperability set-up
VICINITY Cloud
VICINITY Neighbourhood Manager
Semantic discovery & dynamic configuration agent platform
Gateway API Services
VICINITY Communication Server
VICINITY P2P network
VICINITY Node
VICINITY Node
VICINITY Node
Semantic & dynamic configuration data
Virtual neighbourhood Configuration data
User data
Semantic model & agent configuration storage
Global neighbourhood storage

Figure 2 VICINITY Cloud logical view

2.2. VICINITY Node

The VICINITY Node includes the following essential components:

- *VICINITY Gateway API* for: providing interface to registry, discovery, query data from VICINITY, access IoT objects (reading, writing and performing actions), exposing IoT objects in VICINITY;
- *VICINITY Communication Node* that maintains access in peer-to-peer network;
- *VICINITY Agent* that provides physical access to infrastructures adopted into VICINITY, serving as "translator" between infrastructure specific technology and VICINITY common models and services; *VICINITY Adapter* is the part of *VICINITY Agent*;
- *VICNITY Adapter* that provides access to infrastructures adopted into VICINITY via the VICINITY Agent, by implementing common extended functionality of the VICINITY Adapter and handling all IoT objects available in the underlying infrastructure.

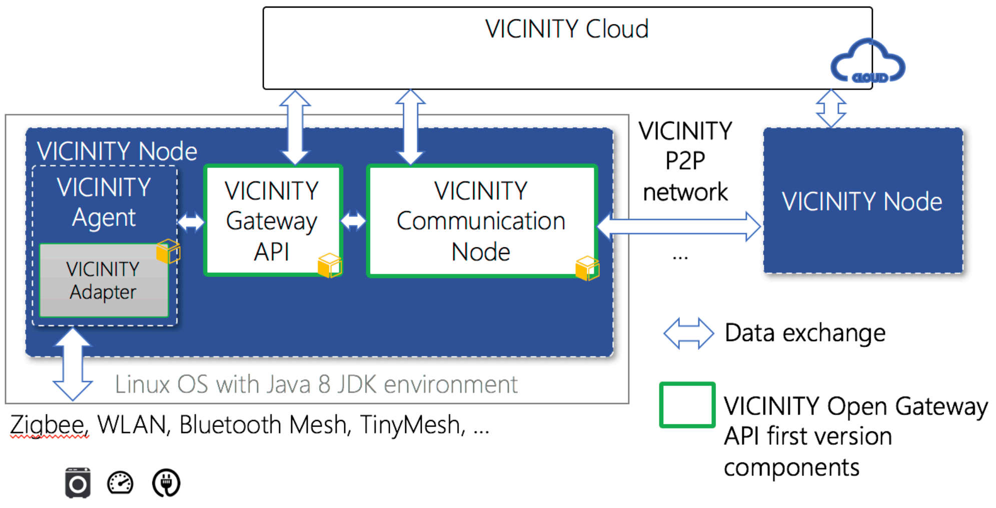
VICINITY Cloud
VICINITY Node
VICINITY Agent
VICINITY Adapter
VICINITY Gateway API
VICINITY Communication Node
VICINITY P2P network
VICINITY Node
...
Linux OS with Java 8 JDK environment
Zigbee, WLAN, Bluetooth Mesh, TinyMesh, ...
Data exchange
VICINITY Open Gateway API first version components

Figure 3 VICINITY Node logical view

2.3. Overview of releases

Based on the VICINITY DoA, the following VICINITY Server Implementation releases have been identified:

- **Release Alpha** includes first version of Open Interoperability Gateway API to support internal development of other server and client components;
- **Release Beta** includes all server components ready for internal VICINITY integration and lab testing;
- **Release RC1** includes updates and fixes from internal VICINITY Integration and lab testing, ready for demonstration & overall evaluation;
- **Release 1** includes updates and fixes from VICINITY demonstration & overall evaluation.

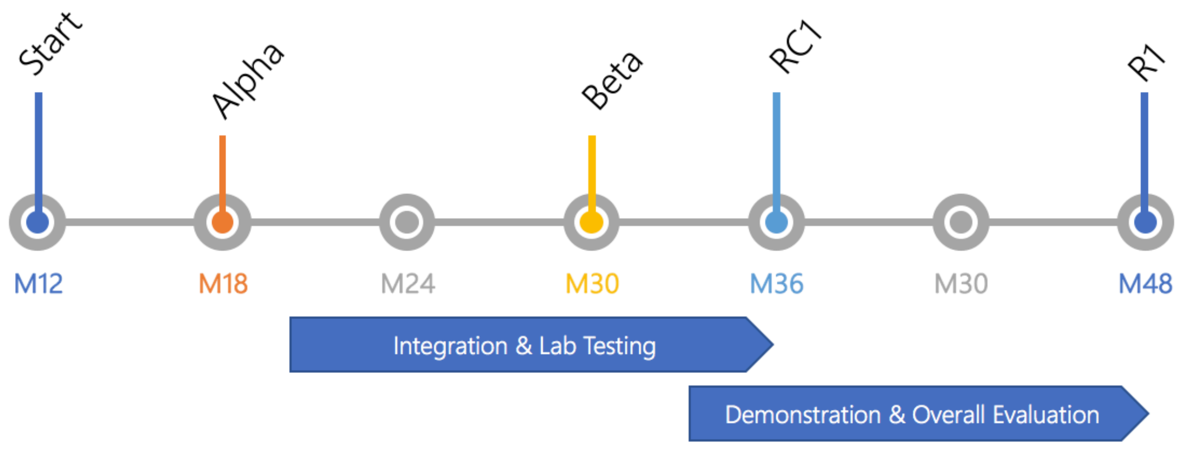

Figure 4 VICINITY Releases time-line

Each of these releases has the detailed goals which are summarized in the following list of releases (Table 1). This table shows only official public releases, e.g. "Beta" release is long 1 year implementation run. Internally this release is broken down in four 3 months long manageable releases. These internal release are continuously adjusted to needs of other WPs such as WP4, WP5 and WP6.

Table 1 List of releases

Name of the release	Goal of the release	Component version	Deliverable	Planned delivery date
Alpha	Open Interoperability Gateway API: • Authentication in VICINITY; • Get and set the current value of a specific property of an (exposed) IoT object; • Perform a specific action upon an (exposed) IoT object (including status); • Get the status of on-going specific actions upon an exposed IoT object;	0.1	D3.3	M18

Name of the release	Goal of the release	Component version	Deliverable	Planned delivery date
	High-available VICINITY server deployment: • P2P Network Node Endpoint; • Gateway API Endpoint translates Gateway API request and responses to/from P2P Network messages;	0.1		M18
Beta	High-available VICINITY server deployment: • P2P Network Server Endpoint; • P2P Network Manager – manages the P2P peer network including P2P network authentication and global white/black list management; • Authorization Services; • Encryption Services; • Transaction Module;	0.5	D3.1	M30
	Web-based VICINITY neighbourhood manager: • Search for IoT objects, users, organizations through; • Management of social network of organizations and IoT objects; • Management sharing access rules; • VICINITY Nodes configuration management; • Consent(s) and terms of conditions management; • Users and organizations management • User notifications; • User auditing; • Authentication and users authorization;	0.5	D3.2	M30
	Open Interoperability Gateway API: • Register / Expose new IoT objects; • Get the list of available interaction patterns (i.e., properties, actions and events) of an IoT object along with each value constraints and units; • Subscribe / Unsubscribe / Listen to events issued by a consumed and exposed IoT object; • Separation of user plain; • Register as VICINITY Gateway API; • Discover IoT objects that "match" to a required search pattern; • Process configuration updates of exposed IoT objects from the VICINITY	0.9	D3.4	M30

Name of the release	Goal of the release	Component version	Deliverable	Planned delivery date
	Cloud; • Query the VICINITY P2P Network by means of a combination of discovery and access functions, by using SPARQL[1] and the VICINITY Ontology; • Generate semantic descriptions of things' ecosystems; • Generate discovery and consumption plans;			
	Semantic discovery and dynamic configuration services: • Registration discovery of IoT objects: automatic and by request; • Update of IoT objects and agent configuration (enable/disable IoT object, enable/disable IoT object service/action, etc.); • Automatic creation of IoT object model templates from specified external repositories of IoT object descriptors; • Providing common semantic vocabulary to describe IoT objects; • Performing CRUD operations on IoT object models and agent configuration; • Executing semantic queries and look-ups; • Serialization of semantic information into common machine-readable format (e.g. JSON, XML);	0.5	D3.5	M30
RC1	High-available VICINITY server deployment	0.9	D3.6	M36
	Web-based VICINITY neighbourhood manager	0.9	D3.6	M36
	Open Interoperability Gateway API	0.9	D3.6	M36
	Semantic discovery and dynamic configuration services	0.9	D3.6	M36
R1	High-available VICINITY server deployment	1.0	D3.7	M48
	Web-based VICINITY neighbourhood manager	1.0	D3.7	M48

[1] For more details see: https://www.w3.org/TR/sparql11-query/

Name of the release	Goal of the release	Component version	Deliverable	Planned delivery date
	Open Interoperability Gateway API	1.0	D3.7	M48
	Semantic discovery and dynamic configuration services	1.0	D3.7	M48

2.4. Development environment

The overall implementation is steered by the release planning (see 2.3). Implementation of the core VICINITY Components has been divided in 4 publicly available release. Note, that longer releases are divided in smaller 3 months long sub releases to better manage long running implementation tasks. The release plan is reviewed by each WP leader (namely WP3 Client Infrastructures Implementation, WP5 Value-Added Services Implementations, WP6 VICINITY Framework Integration & Lab Testing and WP7 On-Site deployment and Pilot Installations) and updates to align the needs and expectations of WPs. Each public release will be formalized in Release Note, which will be publicly available.

Functionalities planned in each release are broken in small tasks managed by respective partner personnel. Technical issues are discussed on-line in real-time messaging environment SLACK allowing to exchange documents, code snippets and searching the archive. Whole implementation team is using SLACK under **vicinity-h2020.slack.com** group.

The source code is managed under common GitLab platform (supported by UNIKL) where GIT repositories are managed for each implemented core component (https://cpsgit.informatik.uni-kl.de/VICINITY [2]). GIT source code management technology enable partners to manage source code locally on their daily basis and synchronize with common project repositories on GitLab after tasks are ready to integrate. Currently the following repositories are managed:

- *vicinity-open-gateway-api* – stores source code of VICINITY Open Gateway API;
- vicinity-agent – stores source code of VICINITY Agent;
- *vicinity-adapter*-CERTH-LinkSmart – stores source code of VICINITY Adapters for Link Smart;
- *vicinity-adapter*-CERTH-SiteWhere – stores source code of VICINITY Adapters for SiteWhere;
- *vicinity-gateway-api-services* – stores source code of VICINITY Gateway API supporting services for discovery and querying;
- *vicinity-semantic-platform* – stores source code of VICINITY Semantic discovery and dynamic configuration platform;
- *vicinity-communication-server* – stores source code of VICINITY Communication Server;
- *vicinity-neighbourhood-manager* – stores source code of VICINITY Neighbourhood Manager.

[2] Access to repository please contact VICINITY Coordinator.

For each official and unofficial release, the source code will be labelled by special tags with name of the release and the identifier of the deliverable (see Table 1) (e.g. "Alpha", "D3.3"). The labelling the source code enables us to keep track which changes are done in which release.

3. Release note overview

The goal of the Alpha release is as follows:

- The VICINITY Open Interoperability API detailed specification for implementation of the VICINITY Adapters for value-added services and early pilot site integration;
- The VICINITY Open Interoperability API shall be able to:
  - provide what is necessary for getting and setting IoT object properties;
  - execute the actions upon the IoT objects, including action status provisioning;
- The VICINITY Open Interoperability API will provide services to authenticate VICINITY Adapters and IoT objects;
- The VICINITY Communication Node will be able to exchange data over preconfigured P2P network(s) between VICINITY Open Interoperability API.

4. List of Issues implemented

The list of issues that have been implemented in this release are listed in the following table.

Table 2 List of issues

Summary	**Component**	**Status**	**Partner**
Specification of VICINITY Gateway API	VICINITY Gateway API	Done	BVR, UPM
Get the status of on-going specific actions upon an exposed IoT object	VICINITY Gateway API	Done	BVR
Perform a specific action upon an exposed IoT object	VICINITY Gateway API	Done	BVR
Set the value of a specific property of an exposed IoT object	VICINITY Gateway API	Done	BVR
Get the status of on-going specific actions upon an IoT object	VICINITY Gateway API	Done	BVR
Perform a specific action upon an IoT object	VICINITY Gateway API	Done	BVR
Set the value of a specific property of an IoT object	VICINITY Gateway API	Done	BVR
List of IoT objects descriptions	VICINITY Gateway API	Done	BVR
VICINITY Communication Node lifecycle management	VICINITY Communication Node	Done	BVR

Summary	Component	Status	Partner
VICINITY Gateway API lifecycle management (start, stop, restart)	VICINITY Gateway API	Done	BVR
Deployment of a new version of the VICINITY Gateway API	VICINITY Gateway API	Done	BVR
Deployment of a new version of the VICINITY Communication Node	VICINITY Communication Node	Done	BVR
Get the value of a specific property of an exposed IoT object	VICINITY Gateway API	Done	BVR
Get the current value of a specific property of an IoT object	VICINITY Gateway API	Done	BVR
Gateway API Endpoint for the VICINITY Communication Node	VICINITY Communication Node	Done	BVR
Route P2P Network message between Communication Nodes	VICINITY Communication Node	Done	BVR
VICINITY Adapter authentication upon the VICINITY Gateway API	VICINITY Gateway API	Done	BVR
Specification of the VICINITY Gateway API for Consumption interface	VICINITY Gateway API	Done	BVR
VICINITY Communication Node logging	VICINITY Communication Node	Done	BVR
VICINITY Gateway API logging;	VICINITY Gateway API	Done	BVR
Configuration of the VICINITY Gateway API	VICINITY Gateway API	Done	BVR
Configuration of the VICINITY Communication Node	VICINITY Communication Node	Done	BVR
Specify Common VICINITY device description format	VICINITY Adapter	Done	IS
Integrate adapter implementations for the CERTH Gateways	VICINITY Adapter	Done	CERTH
Integrate Adapter and Agent components	VICINITY Adapter	Done	IS
Deployment of a new version of the VICINITY CERTH Adapters	VICINITY Adapter	Done	CERTH

Summary	Component	Status	Partner
Implement configuration of the VICINITY Agent to process with username/password	VICINITY Agent	Done	IS
Implement configuration of the VICINITY Agent to process custom username/password for each IoT object	VICINITY Agent	Done	IS
Implement reusable Agent component configurable for different Adapters	VICINITY Agent	Done	IS
Get the value of a specific property of an exposed IoT object	VICINITY Agent	Done	IS
Set the value of a specific property of an exposed IoT object	VICINITY Agent	Done	IS
Perform a specific action on an exposed IoT object	VICINITY Agent	Done	IS
Get the status of on-going specific actions upon an IoT object	VICINITY Agent	Done	IS
Provide IoT object descriptions in the VICINITY Common format	VICINITY Agent	Done	IS
Integrate Agent and Gateway API	VICINITY Agent	Done	IS
Deployment of a new version of the VICINITY Agent	VICINITY Agent	Done	IS

5. List of bugs fixed

There are no bugs fixed in this release.

6. How to apply the release

6.1. How to get access to release

The VICINITY Open Gateway API and examples of adapters are publicly provided at the VICINITY project site on the following link: http://vicinity2020.eu/vicinity/ .

6.2. How to install release

6.2.1. Installation of VICINITY Open Gateway API version 0.1

In order to install the VICINITY Open Gateway API v 0.1, Maven[3] need to be installed first. Keep in mind that the machine running the Gateway API will also need a functional Java Runtime Environment (at least version 1.8). Please refer to the documentation and instructions on how to download and install these tools at:

[3] Apache Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information.

- http://openjdk.java.net/
- https://maven.apache.org/

After these tools have been installed, it is possible to build the sources, so a runnable jar file is obtained. Afterwards, the Gateway can be configured and run.

6.2.1.1. Building the source codes

In the cloned directory, execute the following command:

```
$ mvn clean compile assembly:single
```

The build can take up to a few minutes, as the Maven downloads necessary libraries. The build should finish without errors (it can however finish with some warnings). After it is done, you will find the runnable jar file in the 'target' directory. You are free to rename the file as needed and place it into a directory, where it will be run. In the next section, you will configure your freshly built Gateway.

6.2.2. Configuration of VICINITY Open Gateway API version 0.1

The VICINITY Open Gateway API source comes largely preconfigured, with configuration file 'GatewayConfig.xml' located in 'config' directory of the cloned distribution. This file has all the parameters described by inline comments and, for most parameters, default values are sufficient. However, there are a few options that are necessary to be adjusted before the first run:

- logging file

```
<configuration>
<!--
… other parameters ...
-->

      <logging>
            <file>
            [absolute or relative path to a writable log file]
            </file>
      </logging>

<!--
… other parameters ...
-->
</configuration>
```

- URL / IP address and port of communication server, if accidentally left set to ‘localhost’, change to:

```
<configuration>
<!--
… other parameters ...
-->

      <xmpp>
            <server>dev_be.bavenir.eu</server>
            <port>5222</port>
      </xmpp>

<!--
… other parameters ...
```

```
-->
</configuration>
```

- URL / IP address and port of an Agent

```
<configuration>
<!--
… other parameters ...
-->

      <api>
            <agentIP>[IP or URL]</agentIP>
            <agentPort>[port number]</agentPort>
      </api>

<!--
… other parameters ...
-->
</configuration>
```

Also, do not forget to open necessary port on your firewall. Port the Gateway listens for HTTP requests can be set by following parameter in configuration file:

```
<configuration>
<!--
… other parameters ...
-->

      <api>
            <port>[port number]</port>
      </api>

<!--
… other parameters ...
-->
</configuration>
```

Be aware that running the software on privileged ports (<1024) needs root's privileges.

6.2.3. Running VICINITY Open Gateway API version 0.1

The single executable jar file contains all necessary libraries and is run like this:

```
$ nohup java -jar [the file].jar &
```

Nohup and ampersand character at the end are necessary to keep the Gateway running even after you log out. Service integration script is coming out soon.

6.2.4. Updating VICINITY Open Gateway API version 0.1

A new release can be pulled from the Git repository whenever available. Use

```
$ git pull
```

in the directory, where you previously cloned the repository. After the pull is complete, proceed with building the source codes the same way, as during initial installation. Copy the newly build Gateway API jar file and replace the existing one with it (don't forget to kill the old process first).

No other steps are usually necessary; however read the README file, as it might mention additional step needed to perform (like adding a parameter into configuration file, etc.). Then run it as before.

7. Documentation notes

For integration notes please see Annex A and B of the D3.3 deliverable.

8. List of source code changes

Part of this release are the source code changes in the following repositories:

- vicinity-agent;
- vicinity-adapter-CERTH-LinkSmart;
- vicinity-adapter-CERTH-SiteWhere;
- vicinity-open-gateway-api.

As described in section 2.4, these repositories are managed on the single GitLab platform. Each change (implementation of a feature or fixing a bug) in the source code are represented by the singular commit with unique identifier. This allows developers to investigate or revert source code changes during bug fixing. The source code of relevant repositories has been labelled by “Alpha” and “D3.3” tag for this release. The following tables (Table 3, Table 4, Table 5, Table 6) of lists the changes made in this release.

Table 3 List of source code commits for vicinity-open-gateway-api repository

```
83b9fe4 Updated to stay in line with updated version of D3.3. Minor
improvements implemented.

45bad36 Gateway API integration with Agent now meets the criteria for D3.3.

2d1b3a2 A new re-worked messaging protocol implemented. Services now
compliant with D3.3.

f64b4d0 More refactoring.

5e70336 Forgotten log files deleted...

29e16e3 Minor refactoring and fixes.

3957afc Ability to retreive the property of a remote object implemented.

e5370a1 README.md edited online with Bitbucket

4764def README.md edited online with Bitbucket

f3a933b README.md edited online with Bitbucket

6fe67b0 Maven configuration modified to produce nice builds from terminal.
Git repository cleaned of unnecessary forgotten stuff (log files, previous
builds, etc.).

dbe9c6b README.md edited online with Bitbucket

083b123 README.md edited online with Bitbucket

6e65f84 README.md edited online with Bitbucket
```

```
845f737 README.md edited online with Bitbucket

9f4798f README.md edited online with Bitbucket

8524a70 Gateway API ready for basic integration with Adapters and Objects.

cd6d893 HTTP authentication implemented, REST API and XMPP Controller
integrated, GET /objects returns real values.

bd0d9c6 README.md edited online with Bitbucket

24187cf README.md edited online with Bitbucket

2da4c0b README.md edited online with Bitbucket

e1811fd README.md edited online with Bitbucket

d7f9ad8 README.md edited online with Bitbucket

932d22f README.md edited online with Bitbucket

9cc1854 README.md edited online with Bitbucket

b0c4b24 README.md edited online with Bitbucket

e04dded README.md edited online with Bitbucket

ca628aa README.md edited online with Bitbucket

bb19b51 README.md edited online with Bitbucket

7385dd7 README.md edited online with Bitbucket

d1a560c README.md edited online with Bitbucket

3ac269f README.md edited online with Bitbucket

0ecaa45 Full implementation of Gateway API stubs.

3bfc22e GET and DELETE API stubs are in line with specification.

d8ff4f5 Improved connection manager for XMPP communication.

42a3f4d New connection manager for multiple XMPP connections added.

d085784 Preliminary API stubs created.

c6bbde2 Initial commit by sulfo
```

Table 4 List of source code commits for vicinity-agent repository

```
b2d861a added get action status in testing adapter and agent

bbf6d7a added get action status in testing adapter and agent

eeb4cf3 improved angent spec, added stub for action task status

54705c5 minor aesthetic changes in testing adapter output
```

f5def3d added testing event listener on adapter and testing event generator as stubbed event producer, just for event consumption integration test

c232428 redesign of agent configuration of internal id mapping from vicinity into adapter infrastructure to accept specific login/password per each iot thing. refactored startup sequence and shutdown hook to use gtw api to login/logout agent and iot things into p2p network. now, real login/logout call of gtw api is omitted (needs physical integration)

5cde0cd redesigned core for aau adapter prototype

590cde7 removed no more used adapter module and aau adapter prototype, which will be redesigned

8b4b582 implemented execute action endpoints in testing adapter and agent implementation to execute action at endpoints specified in thing descriptions

606fc8e working on dynamic agent configuration. testing adatper to test all possible endpoints including custom read/write implemented. startup and shutdown agent sequences prepared, now with fake configuration. integrated on both agent and adapter: get/set property

2ed43f1 added thing description basic processor

6c89f87 moved to simple jar application instead of jetty war .. run script updated to be able to start and stop agent correctly, including shutdown hook in server application

06bef92 prepared skeleton for AAU adapter

2461ec4 implemented CERTH SiteWhere adapter test for new sensors and execute action example

b8cf007 preparation for object action execution

78b66e3 gorenje update

42bce93 external config injected via bin/run.sh as system property

22e4de6 added external configuration

fee76c1 external config, dynamic reconfiguration introduced

4c25d19 local notebook checkout

059f2ff implemented example call to /login/ service and /objects/ service with credentials to test the inter-component communication

6ff491f configuration update

367e06f added CERTH LinkSmart adapter handler

1d7915f upadted project infrastructure, separated agents and adapters, agent is now generic module working with specific adapter. Added AgentConfig for VICINITY/Infrastructure object id mapping and configuring agent adapter implementation. Made adapter as standalone service. Implemented AgentAdapter interface to define Adapter API. Implemented first method to get properties of the object. Implemented adapter prototype for CERTH SmartHome infrastructure.

```
83a9f31 client hacked to accept any certificate including invalid

8a272ac added trivial REST client for on demand testing/debug purposes

06733c1 init
```

Table 5 List of source code commits for vicinity-adapter-CERTH-LinkSmart repository

```
19fdc5b First commit of CERTH's Linksmart adapter.

c51474d Update README.md

5bb9f4b Update README.md

c622c23 add README
```

Table 6 List of source code commits for vicinity-adapter-CERTH-SmartWhere repository

```
37cfc3a Fix for action write_links.

f078d53 First commit of CERTH's SiteWhere adapter.

9820483 Update README.md

a368bc7 add README
```

9. Conclusions

The D3.3 - VICINITY Open Gateway API first version is the first of the official VICINITY Core components releases (2.3) – Alpha release – which provides critical functionality to exchange IoT objects properties and performing actions on IoT objects over VICINITY peer-to-peer network. The next official release – Beta release - will include full implementation of the VICINITY Core Components (2.3) ready for laboratory testing and for pilot site validation.

The next release will include the implementation of the registry services, discovery and query services including integration to other VICINITY Core components such as full featured VICINITY Communication Server, Semantic discovery & dynamic agent configuration platform and VICINITY Neighbourhood Manager.

Annex A VICINITY Open Gateway API Integration manual

After installing and configuring the VICINITY Open Gateway API following to the configuration and integration, work needs to be done for:

- Registering the VICINITY Agent in VICINITY;
- Registering IoT objects and setup communication channels;
- Implementing authentication of the VICINITY Agent in VICINITY;
- Implementing authentication of IoT objects in VICINITY;
- Reading of property of IoT object(s) through VICINITY;
- Writing of property of IoT object(s) through VICINITY;
- Performing action upon IoT objects in VICINITY.

Annex A.1 Register VICINITY Agent in VICINITY

In the current version, registration of the the VICINITY Agent in VICINITY needs to be performed by the VICINITY team manually. Please send your enquiry through the official VICINITY Project contact form: http://vicinity2020.eu/vicinity/contact.[4]

Annex A.2 Register IoT objects and setup communication channels

In the current version, registration of the VICINITY Agent in VICINITY needs to be performed by the VICINITY team manually. Please send your enquiry through the official VICINITY Project contact form: http://vicinity2020.eu/vicinity/contact.[5]

Annex A.3 Implement authentication of VICINITY Agent in VICINITY

Concept:

As the VICINITY communication infrastructure relies on XMPP protocol to transport messages across the network, the Agent (or any object utilizing Gateway API) authenticates against the XMPP server with provided XMPP credentials.

The authentication occurs in two stages:

1. The Agent tries to authenticate by using standard HTTP Basic / Digest schema (actual method is set in Gateway configuration file) by calling any service on Gateway API and providing credentials in headers.

2. These credentials are then extracted and used by the Communication Node so that to try to establish a connection with the XMPP server if it does not exist yet, or by verifying the password if the connection does exist, already. A success of this action determines the resulting code to be sent back to the client, making the service request (along with requested data).

4 Note that support to external partner is limited at this stage of VICINITY Project.

5 Note that support to external partner is limited at this stage of VICINITY Project.

From this *modus operandi* it should be clear that the Gateway acts as a “translator” between the stateless REST API services and the stateful Communication Node for the XMPP network. The otherwise beneficial “statelessness” of the Gateway API services however comes with a problem when an Agent only wants to listen for incoming requests, without requesting anything from the XMPP network (but the connection is not established, unless the Agent requests something first, therefore other Nodes cannot see it, and the request can never arrive).

To overcome this issue, two extra endpoints were created, one for login and one for logout. To make things a little clearer, these are not mandatory to use. The connection is established automatically when (and maintained ever since) any authenticated request to the Gateway API is performed. These endpoints just “force” the Communication Node to establish a connection even when there is no real request to be performed.

Reference endpoint:

```
GET http://[Gateway API URL or IP]/api/objects/login
GET http://[Gateway API URL or IP]/api/objects/logout
```

Example of call:

```
$ curl -i -H "Authorization: Basic dXNlcjA6dXNlcjA="
"http://localhost:8181/api/objects/login"
```

Returns "Login successful" or "404 Unauthorized"

Annex A.4 Implement authentication of IoT objects in VICINITY

All technical and conceptual details from Annex A.3 apply also to authentication of IoT objects such as devices and services as the Object and Agent authenticate the same way.

Annex A.5 Reading of property of IoT object through VICINITY

Concept:

After this endpoint is called on the Gateway API, a XMPP message is sent to the Object defined by {oid}, with a request to send back a status of Property defined by {pid}. The message is received by the respective Communication Node module on the remote Gateway (or on the same one, if given object is not that far) and the requested operation is called on Agent, to which the Object is connected.

Response is then sent back over the XMPP network and returned to the original caller.

Reference endpoint:

```
GET http://[Gateway API URL or IP]/api/objects/{oid}/properties/{pid}
```

Example of call:

```
$ curl -i -H "Authorization: Basic dXNlcjA6dXNlcjA="
"http://localhost:8181/api/objects/0d485748-cf2a-450c-bcf6-
02ac1cb39a2d/properties/PowerConsumption"
```

Returns a JSON with a value of `PowerConsumption` property of an object with ID 0d485748-cf2a-450c-bcf6-02ac1cb39a2d.

Annex A.6 Writing of property of IoT object through VICINITY

Concept:

In a very similar manner to A.5, a new Property value (defined by {pid}) of Object (defined by {oid}) in JSON format is propagated through the XMPP network and the resulting response code is propagated back.

Reference endpoint:

```
PUT http://[Gateway API URL or IP]/api/objects/{oid}/properties/{pid}
```

Example of call:

```
$ curl -v -H "Content-Type: application/json" -H "Authorization: Basic
dXNlcjA6dXNlcjA=" -X PUT --data '{"value": 100}'
http://localhost:8181/api/objects/0729a580-2240-11e6-9eb5-
0002a5d5c51b/properties/PropertyToChange
```

Annex A.7 Performing action on IoT objects in VICINITY

Concept:

Request to perform an Action is propagated through the XMPP network to the respective Agent/Object. As the remote Agent processes the request, it generates a new Task ID (the Task is performing the requested Action). This ID is then propagated back. As it can take a while to actually complete the Task, the status (and the potential result) can be accessed with additional request to the second endpoint.

Reference endpoint:

First

```
POST http://[Gateway API URL or IP]/api/objects/{oid}/actions/{aid}
```

then

```
GET http://[Gateway API URL or
IP]/api/objects/{oid}/actions/{aid}/tasks/{tid}
```

Example of call:

```
$ curl -v -H "Content-Type: application/json" -H "Authorization: Basic
dXNlcjA6dXNlcjA=" -X POST --data '{"value": true}'
http://localhost:8181/api/objects/0729a580-2240-11e6-9eb5-
0002a5d5c51b/actions/SomeAction
```

Returns a response with Task ID - {tid}.

```
$ curl -v -H "Authorization: Basic dXNlcjA6dXNlcjA="
http://localhost:8181/api/objects/0729a580-2240-11e6-9eb5-
0002a5d5c51b/actions/SomeAction/tasks/ca43b079-0818-4c39-b896-699c2d31f2db
```

Returns a task status in JSON format.

Annex B VICINITY Open Gateway API version 0.1 interface specification

Please see separate document VICINITY_D3.3_Open_Interoperability_Gateway_API_AnnexB_v0.1.zip or http://vicinity2020.eu/vicinity/content/vicinity-gateway-api-specification .

Annex C VICINITY Agent version 0.1 source code

Please see separate document VICINITY_D3.3_Open_Interoperability_Gateway_API_AnnexC_v0.1.zip or http://vicinity2020.eu/vicinity/content/implementation.

Annex D VICINITY LinkSmart adapter version 0.1 source code

Please see separate document VICINITY_D3.3_Open_Interoperability_Gateway_API_AnnexD_v0.1.zip or http://vicinity2020.eu/vicinity/content/implementation.

Annex E VICINITY SiteWhere adapter version 0.1 source code

Please see separate document VICINITY_D3.3_Open_Interoperability_Gateway_API_AnnexE_v0.1.zip or http://vicinity2020.eu/vicinity/content/implementation.