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 D4.2

VICINITY Agent and Auto-Discovery platform

Work Package:	**WP4 – VICINITY Client Infrastructures Implementation**
Task(s):	**T4.2 - VICINITY Agent and Auto-Discovery platform**
Lead Beneficiary:	**IS**
Due Date:	**30 June 2018 (M30)**
Submission Date:	**29 June 2018 (M30)**
Deliverable Status:	**Final**
Deliverable Type[1]:	**R/DEM**
Dissemination Level[2]:	**PU**
File Name:	**VICINITY-D4.2-Agent-and-Auto-Discovery-platform-v1.0.pdf**

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

VICINITY Consortium

No	Beneficiary		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	HITS	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.

[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**
Kostelnik	Peter	IS	peter.kostelnik@intersoft.sk
No Co-authors			

Reviewers List

List of Reviewers (in alphabetic order)				
No	**Surname**	**First Name**	**Beneficiary**	**Contact email**
1.	Hovstø	Asbjørn	HITS	hovsto@online.no
2.	García-Castro	Raúl	UPM	rgarcia@fi.upm.es
3.	Saleiro	Mário	ENERC	m.saleiro@enercoutim.eu

Revision Control

Version	Date	Status	Modifications made by
0.1	5.6.2018	Initial Draft	Peter Kostelnik (IS)
0.2	5.6.2018	Added release notes	Peter Kostelnik (IS)
0.3	12.6.2018	Added content required by new deliverable template, corrections following the reviewer's comments.	Asbjørn Hovstø (HITS), Peter Kostelnik (IS)
0.4	19.6.2018	Switched to actual deliverable template. Updated according to reviewer's comments.	Asbjørn Hovstø (HITS), Raúl García-Castro (UPM), Peter Kostelnik (IS)
0.5	26.6.2018	Updated according to reviewer's comments.	Asbjørn Hovstø (HITS), Raúl García-Castro (UPM), Mário Saleiro (ENERC), Peter Kostelnik (IS)
0.6	28.6.2018	Quality Check, Final Draft Reviewed	Asbjørn Hovstø (HITS), Raúl García-Castro (UPM), Mário Saleiro (ENERC), Peter Kostelnik (IS)
1.0	29.6.2018	Submission to the EC Update of executive summary	Christoph Grimm (UNIKL)

Executive Summary

This deliverable summarizes the outcomes of task T4.2, "VICINITY Agent and Auto-Discovery Platform. The platform is part of the VICINITY client node. Main purpose of the VICINITY Agent and Auto-Discovery Platform is to easily integrate any infrastructure able to expose the IoT objects. All developments were finished in time.

The platform enables to:

- discover the IoT objects,
- to expose these objects to VICINITY in a semantically interoperable way
- to interact with IoT objects in a unified way
- to automatically (re)configure all mappings and interaction patterns for all discovered IoT objects

The functionality of dynamic auto-configuration consists of the client-side components described in this deliverable and the twin components at VICINITY cloud described in D3.5 Semantic discovery and dynamic configuration services. To understand the whole dynamic auto-configuration process, both the client and the cloud-side components must be checked. The client-side implements the full discovery and auto-configuration process. The cloud-side serves mostly as the storage and search engine for IoT objects. Both components were designed and implemented together.

Table of Contents

List of Figures

European Commission | Horizon 2020 European Union funding for Research & Innovation
IoT European Platforms Initiative

List of Definitions & Abbreviations

Abbreviation	Definition
EC	European Commission
EU	European Union
IoT	Internet of Things
GIT	Version control system for tracking changes of computer files
GitHub	Software platform managing GIT repositories
HTTP	Hypertext Transfer Protocol
REST	REpresentational State Transfer, an architectural style that define set of constraints and properties based on HTTP.
P2P	Peer to peer

1. Introduction

1.1. Context within VICINITY

Deliverable D4.2 is based on requirements specified in WP1, semantic models delivered within WP2 and was designed and implemented with very tight cooperation with the cloud-side delivered by WP3. The design and development of components delivered by D4.2 was tightly coordinated with design and development of VICINITY Adapters delivered by D4.1.

Important is the mutual influence with work in WP5 (Value-Added Services), WP6 (Integration and Testing) and WP7 (Pilot Deployment and Installations). Output of this deliverable is important input into all three mentioned work packages, and also many requirements that arose during the progress in these work packages influenced (and will further influence) the design, implementation and functionality of client-side component, documented in this deliverable. The relationships are illustrated in Figure 1.

Relation to other deliverables

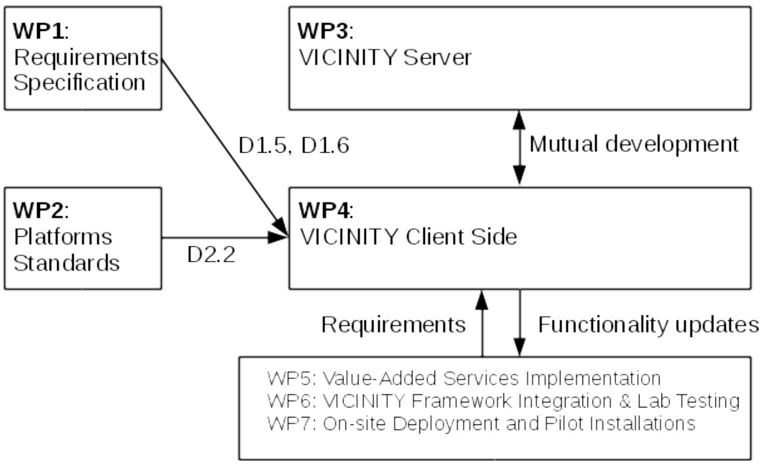
WP1: Requirements Specification
WP3: VICINITY Server
D1.5, D1.6
Mutual development
WP2: Platforms Standards
D2.2
WP4: VICINITY Client Side
Requirements
Functionality updates
WP5: Value-Added Services Implementation
WP6: VICINITY Framework Integration & Lab Testing
WP7: On-site Deployment and Pilot Installations

Figure 1 Deliverable context

- D1.5 VICINITY technical requirements specification summarized the basic requirements identified for this component
- D1.6 VICINITY architectural design describes the position of this component within the VICINITY architecture
- D4.1 Set of open sample VICINITY gateway adapters describes the architecture of Adapters, work in this deliverable was designed, coordinated and implemented together with work in T4.2
- D3.4 Open Interoperability, Gateway API describes the VICINITY Gateway implementation, which serves as the proxy to VICINITY cloud and P2P network
- D2.2 Detailed Specification of the Semantic Model describes the VICINITY ontology, which design was coordinated with specification of VICINITY Common Thing Description format
- D3.5 VICINITY semantic discovery and dynamic configuration services describes the cloud-side twin, which was designed and developed together with this client-side component

1.2. Objectives in Work Package WP4 and Task T4.2

The goal of WP4 - VICINITY Client Infrastructures Implementation is to provide an easy-to-use framework for integration of IoT infrastructures. This includes the design and implementation of VICINITY Gateway Adapters (T4.1), Agent and Auto-Discovery platform (T4.2), Security Services (T4.3) and continuous upgrades of client-side components (T4.4).

This deliverable describes the outcomes of task T4.2. The main goal of this task is to provide the so-called Agent component responsible for auto-discovery and auto-configuration of client nodes, including the maintenance of semantic repository of IoT descriptors.

Deliverable D4.2 Agent and Auto-Discovery Platform is dedicated mainly to infrastructure integrators and developers. The full platform documentation contains all necessary technical and implementation details for integrating the new infrastructure. This documentation serves as the baseline information and guideline for integration of new infrastructures and pilot applications, within the VICINITY pilots and testing scenarios.

This documentation is especially important for organizations joining VICINITY consortium in Open Calls, to understand the details of integration process and technology. The goal of this documentation is to make the integration of new infrastructure as easy as possible.

Documentation is technical document, thus it requires the knowledge of VICINITY architecture and at least the REST technology, as minimum (but should be sufficient) requirement.

The VICINITY client node enables to easily integrate any infrastructure exposing the IoT objects in (quite) easy way. The purpose of client node is to enable the interoperability – the ability to interact with all IoT objects in the common unified way.

The VICINITY client node (see Figure 2) consists of three parts:

- **Adapters**. The translators between the language of specific infrastructure and VICINITY. This translation is the lowest and backbone level of interoperability in VICINITY. The resulting effect is the ability of VICINITY to interact with all integrated infrastructures and their IoT objects in the unique, common way. Adapters expose and provide access to IoT objects presented in underlying infrastructure.
- **Agent**. Extends the functionality of Adapter. Agent translates between internal identifiers of Adapter IoT objects (infrastructure identifiers) and common VICINITY identifiers. Agent holds this mapping together with credentials of IoT objects into P2P network. Agent provides common functionalities, such as automatic IoT objects discovery, automatic (re)configuration of mappings for all discovered IoT objects, automatic log in/out of objects into P2P network, automatic configuration of event channels. Generally, Agent provides services for Adapter developers to easily integrate the underlying infrastructure and to easily interact with IoT objects within VICINITY.
- **Gateway API** providing the access to VICINITY P2P network and VICINITY cloud.

The VICINITY Agent and Auto-Discovery platform was designed to manage multiple VICINITY Agents at the same time. The purpose of this multi-tenant approach was to enable to run many Client Nodes at the same software installation. Normally, for each Client node, there are three required software components (Gateway, Agent, set of Adapters), which must be configured and deployed. In a situation where a VICINITY user needs to run many Client nodes at the same time, the management of running deployed software (three components per Client node) becomes critical. For example, this is the typical scenario for cloud solutions. To make process of deployment of new Client node easy, Agent Platform enables to run multiple Agents in the single software installation. The whole architecture of VICINITY Client node is illustrated in figure 2.

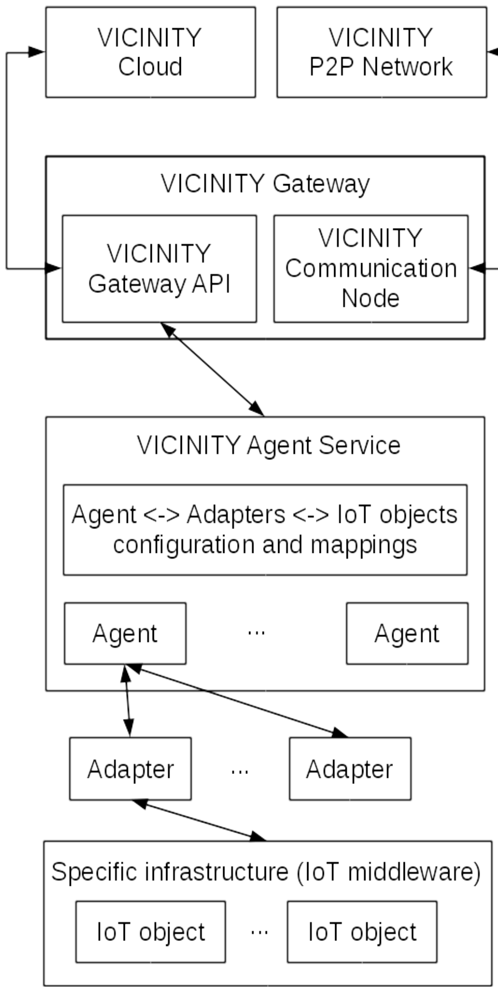

Figure 2 VICINITY Client Node

The Agent Platform enables the asynchronous re-configuration of the included components, so the new Agent (for new client node) may be added on the fly, any of running Agents or also the Adapters may be reconfigured on the fly, without need for restarting whole platform.

This document includes only the Release Notes for the Agent and Auto-Discovery platform. The task T4.2 formally ends in M30, however, the development of the platform still continues to support the implementation of pilot applications. The actual list of features will be continually extended, driven by specific requirements on the fly, as pilot applications will be developed.

The full actual documentation including the code, history of commits, more detailed description and all technical details is available at GitHub:

https://github.com/vicinityh2020/vicinity-agent

2. Release notes

This section contains the overview of Agent and Auto-Discovery Platform functionality.

2.1. Current features

This section contains the list of all implemented features according to all relevant requirements defined in D1.5 VICINITY technical requirements specification and according to architecture specification defined in D1.6 VICINITY architectural design.

- VICINITY Common Thing Description format
  - Design and specification of VICINITY Common Thing Description format
  - Alignment of VICINITY Common Thing Description format with VICINITY ontology
  - Parser of VICINITY Common Thing Description format
  - Syntactic and structural validator of VICINITY Common Thing Description format, the same validator was used in Semantic discovery and dynamic configuration services (D3.5)
- Auto-configuration core
  - Automatic configuration of agent – adapters hierarchy into internal representation
  - Configuration of all structural mappings
  - On-the-fly re-configuration of agent (add new, update existing)
  - On-the-fly re-configuration of adapter (update existing)
- Auto-discovery
  - Passive discovery of IoT objects – depending on configuration, Agent platform asks for the list of exposed objects from Adapters
  - Active discovery of IoT objects – at any time, Adapter may announce the new list of exposed objects
- IoT objects auto-configuration (triggered by IoT objects auto-discovery)
  - Automatic configuration of IoT object mappings in Agent platform, the last active configuration of related Adapter is consulted with semantic model
  - Computing difference between last active and current list of objects
  - Application of IoT object description contract violation rules
  - Automatic update of semantic models for actually exposed IoT objects (CRUD), semantic repository always contains the up-to-date set of IoT object models.
  - Update of mappings between VICINITY unique identifiers and local infrastructure specific identifiers
  - Persistence of VICINITY credentials for all active IoT objects
  - Update of endpoints for IoT object interaction patterns
- Interaction patterns
  - IoT object properties
    - read the property of remote IoT object
    - set the property of remote IoT object
    - read the property of local IoT object
    - set the property of local IoT object
  - IoT object actions
    - read the status of remote IoT object action
    - execute of remote IoT object action
    - read the status of local IoT object action
    - execute of local IoT object action

    - IoT object events
        - Automatic opening event channels and subscriptions to event channels
        - Dynamic opening event channels and subscriptions to event channels by request
        - Production of IoT object event
        - Consumption of IoT object event
- Configuration utilities

2.2. Planned features

Agent and auto-discovery platform will be further maintained and extended with new features, following the requirements from other software components and the pilot site applications. Development will be coordinated using selected issue tracking tools.

- Further improvements and optimization
- The “on-the-fly” requirements for pilot application implementation
- Updates for the identified issues

3. Conclusions

This deliverable covered the outcomes of the task T4.2 VICINITY Agent and Auto-Discovery Platform. The full discovery and configuration process consists of client part and cloud part. This deliverable covers the client part, which fully implements the auto-discovery and auto-configuration process. The cloud part serves mostly as the storage and search engine. The cloud part is described in T3.5 Semantic discovery and dynamic configuration services. To fully understand the process, the both deliverables should be checked.

The full actual documentation including the code, history of commits, more detailed description and all technical details is available at GitHub:

https://github.com/vicinityh2020/vicinity-agent