IoT-NGIN

PPROVAL

D5.4
Enhancing IoT Data Privacy & Trust (Update)

DRAFT - -

WORKPACKAGE	WP5	**PROGRAMME IDENTIFIER**	H2020-ICT-2020-1
DOCUMENT	D5.4	**GRANT AGREEMENT ID**	957246
REVISION	V1.0	**START DATE OF THE PROJECT**	01/10/2020
DELIVERY DATE	30/09/2022	**DURATION**	3 YEARS

This project has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No 957246

DISCLAIMER

ACKNOWLEDGEMENT

This document is a deliverable of IoT-NGIN project. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 957246.

The opinions expressed in this document reflect only the author's view and in no way reflect the European Commission's opinions. The European Commission is not responsible for any use that may be made of the information it contains.

PROJECT ACRONYM	IoT-NGIN
PROJECT TITLE	Next Generation IoT as part of Next Generation Internet
CALL ID	H2020-ICT-2020-1
CALL NAME	Information and Communication Technologies
TOPIC	ICT-56-2020 - Next Generation Internet of Things
TYPE OF ACTION	Research and Innovation Action
COORDINATOR	Capgemini Technology Services (CAP)
PRINCIPAL CONTRACTORS	Atos Spain S.A. (ATOS), ERICSSON GmbH (EDD), ABB Oy (ABB), NETCOMPANY-INTRASOFT SA (INTRA), Engineering-Ingegneria Informatica SPA (ENG), Robert Bosch Espana Fabrica Aranjuez SA (BOSCHN), ASM Terni SpA (ASM), Forum Virium Helsinki (FVH), ENTERSOFT SA (OPT), eBOS Technologies Ltd (EBOS), Privanova SAS (PRI), Synelixis Solutions S.A. (SYN), CUMUCORE Oy (CMC), Emotion s.r.l. (EMOT), AALTO-Korkeakoulusaatio (AALTO), i2CAT Foundation (I2CAT), Rheinisch-Westfälische Technische Hochschule Aachen (RWTH), Sorbonne Université (SU)
WORKPACKAGE	WP5
DELIVERABLE TYPE	REPORT
DISSEMINATION LEVEL	PUBLIC
DELIVERABLE STATE	FINAL
CONTRACTUAL DATE OF DELIVERY	30/09/2022
ACTUAL DATE OF DELIVERY	30/09/2022
DOCUMENT TITLE	Enhancing IoT Data Privacy & Trust (Update)
AUTHOR(S)	Henri Kinnunen (ABB) (ed.), Juuso Autiosalo (AALTO), Helmi Hirvelä (ABB), Jonathan Klimt (RWTH), Yki Kortesniemi (AALTO), Dmitrij Lagutin (AALTO)
REVIEWER(S)	Jonathan Klimt (RWTH), Yki Kortesniemi (AALTO), Terpsi Velivassaki (Synelixis)
ABSTRACT	SEE EXECUTIVE SUMMARY
HISTORY	SEE DOCUMENT HISTORY
KEYWORDS	cyber security, privacy, distributed ledger technology, interledger, self-sovereign identity, decentralised identifier, verifiable credential, digital twin, interoperability, ontology, semantic twin

Document History

Version	Date	Contributor(s)	Description
V0.1	21/06/2022	AALTO, ABB	Template for new document
V0.2	15/9/2022	AALTO,ABB,RWTH	First full draft for internal review
V0.3	21/9/2022	AALTO,ABB,RWTH	Updated draft based on review comments from AALTO & RWTH
V0.4	26/9/2022	AALTO,ABB,RWTH	Review feedback implemented, order of chapters 3 and 4 changed
V1.0	29/9/2022	AALTO,ABB,RWTH	Finalised based on final quality check

Table of Contents

List of Figures

List of Tables

List of Acronyms and Abbreviations

AAS	Asset Administration Shell
DIB	Decentralised Interledger Bridge
DID	Decentralised IDentifier
DLTs	Distributed Ledger Technologies
DNS	Domain Name System
DPoP	Demonstrating of Proof-of-Possession
DSM	Decentralised State Management
DT	Digital Twin
DTDL	Digital Twins Definition Language
FIB	Flexible Interledger Bridge
FOAF	Friend-of-a-friend
IAA	Identity, Authentication, and Authorisation
IoT	Internet of Things
JWT	JSON Web Token
MLDT	Meta-Level Digital Twin [deprecated]
NGSI-LD	Next Generation Service Interfaces-Linked Data API
OData	Open Data Protocol
SAREF	Smart Applications REFerence ontology
SOSA	Sensor, Observation, Sample, and Actuator
SSI	Self-Sovereign Identities
SSN	Semantic Sensor Network
ST	Semantic Twin
STA	SensorThings API
VC	Verifiable Credentials
WoT-TD	Web of Things Thing Description
W3C	World Wide Web Consortium
ZKP	Zero-Knowledge Proof

Executive Summary

This document focuses on the problems of cybersecurity, *privacy preservation and trust improvement* in the domain of IoT systems and presents the technical approaches to tackle the problems developed in the IoT-NGIN project. Specifically, it reports on the results from the Work Package 5 tasks T5.3-5.

In the deliverable D5.3 [D5.3], the requirements from the use cases in the IoT-NGIN project were identified and analysed to determine the best features and properties for the technical solutions to be developed within the IoT-NGIN Work Package 5. The State-of-the-Art technological solutions in the field of *multi-ledger operations*, *semantic interoperability* practices for *Digital Twins, and Self-Sovereign Identities* were then analysed and, finally, the document then provided a high-level description of the solutions that were to be developed within WP5.

This document now presents a description of the first versions of the solutions. Specifically, the *Decentralised Interledger Bridge (DIB)* has been selected as a solution to fulfil IoT-NGIN requirements for interledger.

Semantic Twins are developed in the IoT-NGIN project as a general solution for adding metadata to Digital Twins. The motivation for building Semantic Twins, semantics, and ontologies used for Semantic Twins, and the details of the Semantic Twin solution are given in this document.

Finally, the details of the two *Self-Sovereign Identity* technologies are explored in this document: *Verifiable Credential based decentralised on-device access control with constrained IoT Devices* and *QR code and GS1 Digital Link based discovery mechanisms*.

The final versions of the solutions and their validation results will be presented in the upcoming deliverable D5.5.

1 Introduction

The expanding use of IoT solutions has enabled many new services, but has also raised a range of new privacy and trust challenges. Ubiquitous IoT makes it possible to have a much more accurate and up-to-date situational awareness, but this can pose major privacy issues to the individuals, whose actions are being observed with this technology. Furthermore, individuals themselves are deploying more IoT devices and are in some cases even making the collected data available to a wider audience to enable new services, but at the same time also potentially raising privacy issues. Finally, for the audience utilising the data, a key question is, which IoT devices and data to trust in this abundance of options.

This document addresses these problems in the context of the IoT-NGIN project using the technologies developed in tasks T5.3-5 of Work Package 5: multi-ledger operations, semantic interoperability practices for Digital Twins, and Self-Sovereign Identities. For each of the technologies, the requirements and the State-of-the-Art of the technology were described in the previous report [D5.3]. The current document now provides a description of the first versions of the solutions, and the final versions and their validation results will be described in the upcoming deliverable D5.5.

1.1 Intended Audience

This document is intended for the following groups of people:

- Technical people interested in IoT systems, decentralised applications, digital identity management, and Digital Twin interactions can find detailed solutions and some initial results in use cases.
- Solution designers and policymakers may find the document helpful to understand what kind of services the different technical solutions enable, which level of trust and privacy protection can be provided, and what standard ways for semantic interoperability are possible.
- Internal users within the IoT-NGIN project can find useful resources on the components or architecture solutions that are being made available in WP5, so that use of developed modules is made easier.

1.2 Relations to other activities

This document describes technical solutions involving interledger, Self-Sovereign Identities, ontologies and Semantic Twins (ST), and can, thus, provide guidelines to other work packages in the project on best practices in these fields. The following IoT-NGIN project documents provide further information about the related project activities, which can be useful to extend the knowledge in this document. Architectural elements used in the IoT-NGIN project are described in Deliverable D1.2 [D1.2]. Deliverable D6.2 [D6.2] describes initial versions of the use case applications and initial testing and evaluation results. The upcoming Deliverable D7.3 [D7.3] will provide intermediate results about Living Labs use cases.

1.3 Document overview

The rest of the document is organised as follows.

Section 2 gives an overview on the discussed technologies and how they interact.

Section 3 defines the concept of a Semantic Twin (called a Meta-Level Digital Twin (MLDT) in the IoT-NGIN proposal), and describes the related solutions.

Section 4 covers the Decentralised Interledger Bridge (DIB) solution.

Section 5 presents two different Self Sovereign Identities solutions based on the use case requirements within the project.

Section 6 describes how the solutions mesh together to provide a comprehensive solution as depicted in the demo being developed.

Section 7 concludes the report.

Annex 1 describes how a Semantic Twin approach is applied in case of powertrain.

2 Overview of IoT data privacy and trust in IoT-NGIN

Much of the cybersecurity and privacy work in WP5 tasks T5.3-5 focuses on the IoT-device Triplet shown in the centre of figure 2.1. The *Triplet* consists of a *real-world entity* (in this case, an IoT device), the *Digital Twin (DT)* that exposes the device's capabilities on the net, and the *Semantic Twin (ST)* that semantically describes the other two. When the real-world entity is something other than an IoT device (e.g. a shopping mall or a person), the Triplet can also be called an Entity Triplet, but in IoT-NGIN the focus is mostly on Triplets with IoT devices.

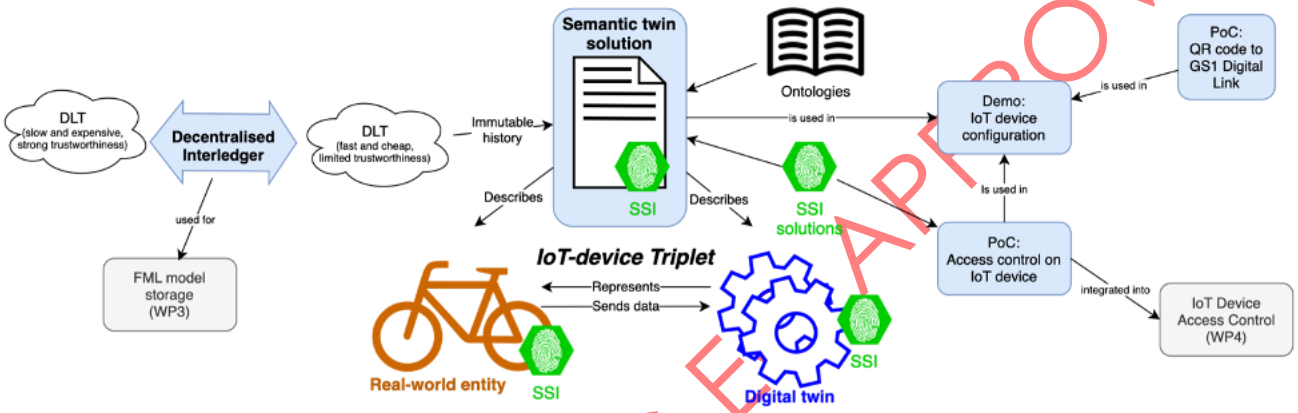

Figure 2.1 - The IoT-device Triplet -related technologies developed in WP5.

To support the IoT-device Triplet, WP5 is developing multiple solutions, as shown in blue in the figure. First, the *Semantic Twin* is a novel concept of providing a structured semantic description of the Triplet. The core element is describing the capabilities of the IoT device and Digital Twin and where they can be accessed. This information can then be complemented with many other types of information, e.g. the licensing of the services and where access could be purchased, information about the validity of the services through, e.g. 3rd-party certification, etc. To make this semantic information as machine-readable and interoperable as possible, the information is organised based on ontologies, particularly *Smart Applications REFerence ontology* (SAREF) ontologies that are aimed for IoT use cases. The Semantic Twin is detailed in Chapter 3.

Another technology being developed is a Decentralised Interledger Bridge that allows us to link distributed ledgers (DLTs) and blockchains with atomic transactions. There are multiple interledger solutions, but most of them only focus on financial transactions or have limitations on the types of DLTs/blockchains they support as described more in detail in Deliverable D5.3 [D5.3]. IoT-NGIN is focusing on a bridging-type interledger, which supports a broad range of ledgers and is agnostic of the transaction type, so it can be used with almost any type of application. Specifically, the work builds on an existing centralised bridging solution, which provides suitable functionality and interfaces, but suffers from the limitations of a centralised solution, namely higher trust requirement on the party running the bridge and lower resiliency. IoT-NGIN is, therefore, developing a decentralised version of the technology, the *Decentralised Interledger Bridge (DIB)* described in Chapter 4, which allows us to overcome the limitations by utilising the same decentralisation approach as the DLTs and blockchains themselves rely on. With the interledger, e.g. the Semantic Twins can now rely on multiple ledgers to provide immutability in a cost-effective manner.

To improve the privacy of the people utilising the Triplet, our work utilises Decentralised Identifiers (DIDs), an identifier technology that follows the Self-Sovereign Identity (SSI) principles. An SSI identity owner should be able to generate and use as many anonymous identifiers as they need to protect their privacy, e.g. to prevent correlation attacks resulting from the same identifier being used in multiple contexts (discussed in Section 3). We also utilise another SSI-technology, Verifiable Credentials (VCs), to carry information about the trustworthiness of different parties (discussed in Section 4) and to implement decentralised access control solutions (Section 5.1). The use of DIDs and VCs has been previously explored mostly in the context of people and organisations, but we are here focusing on their use for things, IoT devices, and the related twins, in order to bring the privacy and trust benefits also to this application area.

To make the use of Semantic and Digital Twins convenient, we are also exploring using digitally signed QR codes and GS1 Digital Links as a convenient and secure way to discover the Twins related to a particular IoT device as detailed in Section 5.2. These types of new usability-oriented solutions are required to enable wide-scale usage of Twin-based solutions.

Finally, to illustrate how these solutions work synergistically, a demo of IoT device configuration is being developed as detailed in Section 6. It will deploy all of the above technologies in the Jätkäsaari Living Lab to demonstrate how we can improve cybersecurity and protect users' privacy in an easy-to-use manner.

3 Semantic Twins

This chapter describes the Semantic Twin solution, whose basic positioning in the IoT-Triplet is shown in Figure 3.1. The following subsections describe the motivation for building Semantic Twins, semantics, and ontologies used for Semantic Twins, and the details of the Semantic Twin solution.

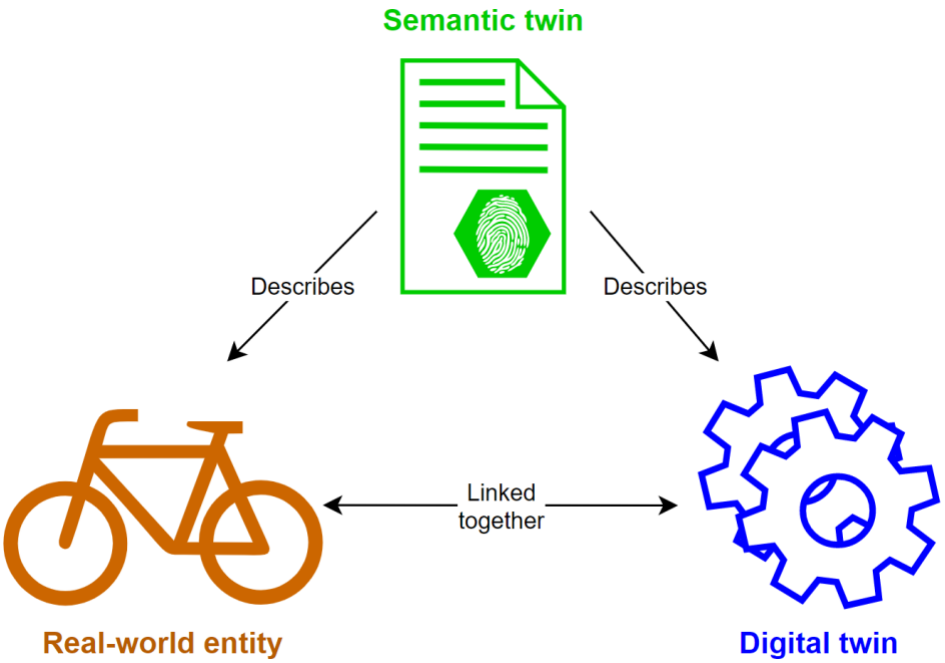

Figure 3.1 - A Semantic Twin describes a real-world entity and its Digital Twin.

3.1 Motivation for Semantic Twins

Recent years have brought us smart entities that consist of a physical entity and its Digital Twin. However, Digital Twins are currently not defined well enough to easily build scalable applications on top of them. Legacy Digital Twins are also missing the basic components needed for data privacy and trust, something the IoT devices themselves also often crave.

The following subsections discuss issues in IoT systems and Digital Twins, lay out requirements for Semantic Twins, and describe the role of Semantic Twins.

3.1.1 Issues in IoT systems and Digital Twins

Legacy IoT devices are configured in a myriad of ways. While this approach has worked well for isolated use cases, it has not enabled IoT devices to act in a properly networked manner. Three important root causes are:

- IoT devices are (in most cases) constrained in technical capabilities (e.g. limited computation capability, communication bandwidth, and power usage).
- IoT devices require a high degree of security and trustworthiness due to being able to create damage in the real world.

- The lack of scalable technical solutions for traversing between the physical and digital worlds (e.g. conveniently accessing sensor data while being physically close to the sensor).

Most of the technical constraints can be overcome with the usage of Digital Twin solutions, but achieving adequate level of security and trustworthiness in a networked environment still requires new solutions. Digital identity solutions may be used to solve some of the trustworthiness issues, but some need other types of arrangements, such as suitable data management architectures.

Digital Twins are virtual counterparts of real-world things. From there on, the definitions diverge according to use case. The Digital Twin concept originated from the product lifecycle management domain in engineering and was adopted as a metaphor for a simulation model that is connected to a real-world machine. Simultaneously, the IoT domain developed concepts and solutions, such as digital agents and sensing technologies, that would later be integrated into the Digital Twin concept. Furthermore, many other digital technologies such as artificial intelligence and augmented reality have been associated with Digital Twins, making the concept fruitful ground for misunderstandings.

For the purpose of this document, we define a Digital Twin as a *collection of software services that are related to a real-world entity*. Some of the software services may be accessible through the public internet, others only in an isolated network and running on local machines. All of these services may provide value for people dealing with the corresponding real-world entity, but there are no conventions on how to deal with these heterogeneous solutions.

3.1.2 The Role of Semantic Twins

Semantic Twins are being developed in the IoT-NGIN project as a general solution for adding metadata to Digital Twins and the real-world entities. Semantic twins differ from Digital Twins in that Digital Twins are complex digital services that can accomplish almost any digital task, whereas Semantic Twins concentrate on meta-level tasks such as identification and description. In other words, Semantic Twins give context and meaning to Digital Twins and real-world entities.

Semantic twins provide information about the services of real-world entities and their Digital Twins in a unified human and machine-readable format. Semantic Twin is a solution that aims to make the integration of Digital Twins and their real-world counterparts more structured and efficient. To achieve this goal, Semantic Twins consist of three main components: Twin ID, twin document, and semantic descriptions, which are further described in section 3.3. Figure 3.2 shows how a Semantic Twin describes the various services that comprise a Digital Twin.

Digital Twins consist of digital services that are related to the real-world entity. These services can be very diverse, such as a cloud-based IoT platform, simulation model, database, or an artificial intelligence agent. These services are also implemented in diverse ways and may be accessible in the cloud or only as local software that is run without internet access. The Semantic Twin needs to be able to provide its services in all of these situations.

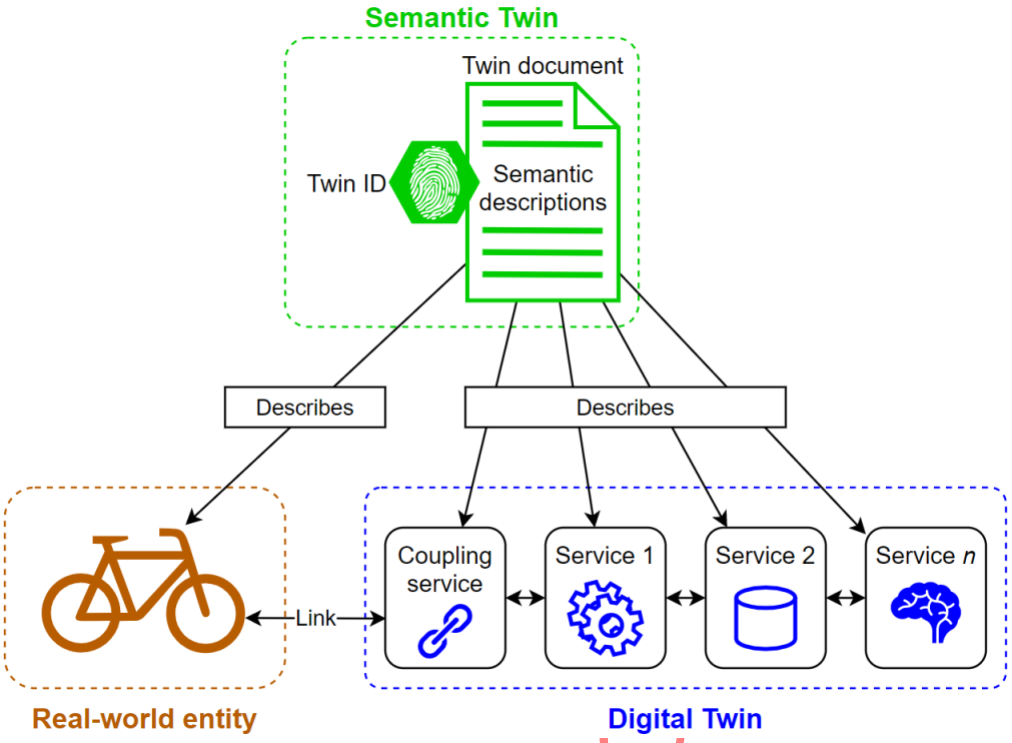

Figure 3.2 - A detailed look into an entity triplet.

A Semantic Twin represents both the real-world entity and the Digital Twin. The Twin ID enables the identification of the Semantic Twin, and therefore, the ST-DT-entity triplet, and this identification may be linked to the real-world entity and Digital Twin services through the descriptions. For example, an external service may access the database service of a Digital Twin via the Twin ID and the semantic description of that service. To achieve scalable machine-readable access to the services of the Digital Twin, the descriptions should follow commonly used semantic vocabularies.

As an example case, twin documents have been used in machine-to-machine communication of a simulated factory, where machines accessed the communication details and relationship descriptions of other machines from their twin documents to fulfil a logistics-related task [Mat2022]. This approach, however, assumes that all parties are trusted, limiting its applicability only to environments to where access is restricted from the outside.

In the long term, Semantic Twins help create a global network of Digital Twins. We call this network the "Digital Twin Web" due to the intended analogy to the "World Wide Web" as further explained in [Aut2021a].

3.2 Description of the Semantic Twin solution

The functional architecture of a system that uses the Semantic Twin solution is shown in Figure 3.3. The twin document is the central component of the Semantic Twin, providing the main body of information. Other components in the green box provide various services for enhancing discoverability and trustworthiness of the solution.

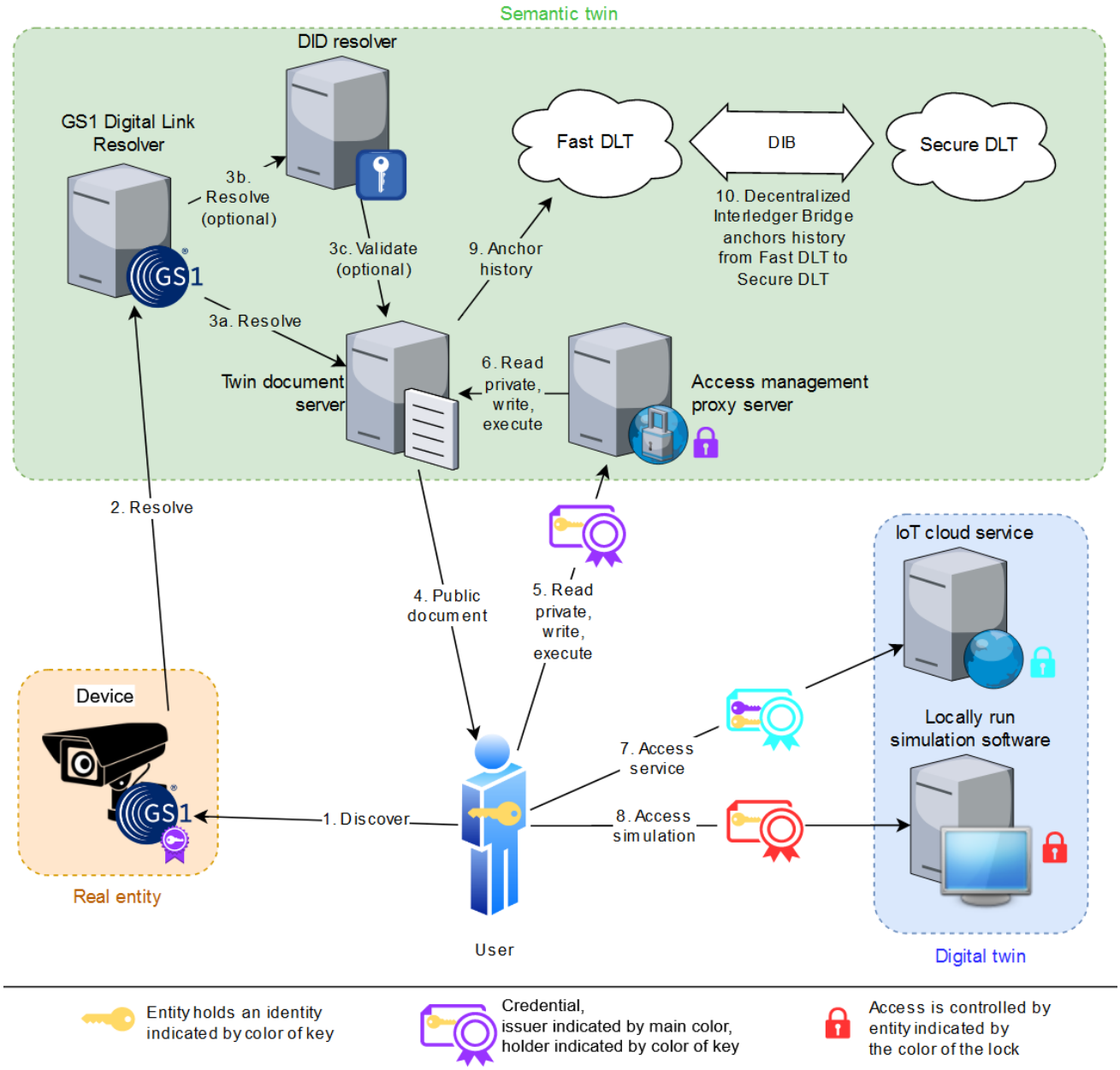

Figure 3.3 - Example composition of a system of systems using different features of STs.

From the user perspective, the Semantic Twin journey starts from (1) the discovery of an identifier, which in this case is the GS1 Digital Link. It can be discovered via a QR code on the physical device or as text string around the internet. The GS1 Digital Link (2) resolves via the Domain Name System (DNS) to a GS1 Digital Link Resolver, which (3a) by default resolves to the twin document server, but may also (3b) resolve to a DID resolver when read with specially made software. The DID can then (3c) provide additional validation for the twin document.

(4) The public part of the twin document is then sent to the user. If the user holds the appropriate credentials, they can (5) read the private part of the twin document and modify

it, and execute operations via an access management proxy server that (6) redirects the requests to the twin document.

The user reads the twin document that describes the methods to access an IoT cloud service and a locally run simulation software. The user decides to (7) access the IoT cloud service with a credential (that was given via delegation). Then the user (8) accesses a simulation on a local environment with a credential that requires no internet access.

(9) The twin documents are hashed and the hash is stored to a fast distributed ledger in short time intervals to anchor the history of the twin document within a small community. (10) On longer time intervals, the hashes are stored (with salt to preserve privacy as discussed in Section 4.1.2) to a more secure ledger via a Decentralised Interledger Bridge (DIB) to provide history verification by the community of the secure ledger.

As demonstrated by the description of the architecture, the main services of the Semantic Twin solution are:

- Provide a description document of the real entity and its Digital Twin services.
- Provide a resolvable ID for the entity triplet.
- Provide validation of the twin document.
- Manage access to the document and potentially to the device and Digital Twin.
- Verify the history of the twin document, in both fast and secure methods.

The three main topics, twin document, discoverability and trustworthiness, and semantic descriptions, of the Semantic Twin solution are further described in the following subsections.

3.2.1 Twin document

A twin document (Digital Twin description document) is a text document that describes a Digital Twin and its real-world counterpart. A twin document is supposed to be the initial source of information about a real-world entity in all use cases. As the document is text-based, any dynamic materials are added as links or interface descriptions.

The distinction between a twin document and a semantic description is that a twin document provides the overall format, and semantic descriptions are the actual contents written in that format. Hence, a twin document is kind of a shell for more detailed information.

We currently use unstandardized formats for twin documents because we have not yet been able to prove that one format fulfils enough requirements to be useful enough. Unstandardized formats can be used in limited experiments and applications, but in the long term, a standardised format is required to achieve most of the benefits of Semantic Twins. Currently, the strongest candidates for twin document format are:

- *Asset Administration Shell* (AAS) [AAS]
- *Web of Things Thing Description* (WoT-TD) [WoT-TD]
- *Digital Twin Definition Language* (DTDL) [DTDL]
- *Next Generation Service Interfaces-Linked Data API* (NGSI-LD) [NGSI-LD]

Those were compared by [Jac2020]. We currently balance between the solutions, but have decided to use JSON-LD as the format of our twin documents. We added support for JSON-

LD to the open-source twin document hosting software "Twinbase" [Twinbase] and developed the Semantic Twin ontology in a format that supports JSON-LD.

The general concept of the twin document was introduced by [Ala2021] and a method to distribute them was introduced by [Aut2021b]. To be clear, the term *twin document* refers to the overall concept and not any specific style of implementation.

3.2.2 Discoverability and trustworthiness

The discoverability and trustworthiness of Semantic Twins are achieved with various identifier/identity and distributed ledger solutions. Discoverability is implemented with a "Twin ID" concept, whereas trustworthiness is a more complex combination of Twin ID and other solutions, with a special focus on distributed ledgers to provide immutable history.

The term "Twin ID" refers both to the identifier and identity solutions of Semantic Twin systems. We use both of these terms because they are conceptually different and have different technical implementations, but still either of those might be needed depending on the use case. Some Semantic Twin use cases may require a full-fledged identity solution with advanced features, such as verifiable credentials, whereas other cases might require anonymity and therefore use temporary identifiers for privacy reasons. Also, depending on the use case, separate IDs may be needed for each Digital Twin service as well as the real-world entity. In addition to one-way linking from a Semantic Twin to a Digital Twin, it may also be beneficial to implement a bidirectional linking. For example, a Digital Twin service may update its own description in the Semantic Twin to keep it up to date, or a Digital Twin service may use the credentials administered by the Semantic Twin to access restricted information in other Digital Twins.

Three main methods of Twin IDs have been identified: a plain URL, a GS1 Digital Link, and DIDs of different methods. Twin ID technologies are still under development, and we use simplified solutions to get started immediately. A baseline solution for a Twin ID is to use a dedicated URL as an identifier for a twin so that the URL is redirected to the corresponding twin document. This however does not allow more granular features that the use of GS1 Digital Links and DIDs enable. GS1 Digital Links enable several redirects from one URL, and DIDs enable e.g. short-lived identifiers and the assignment of a verifiable credential that can be used to access various services. However, simple URL redirections are readily available on the internet for free, whereas GS1 Digital Links require hosting or paying for a server, and DIDs require that the user holds and uses cryptographic keys correctly. These may not be obstacles for organisations with strong research and development capabilities, but may hinder adoption in more production-oriented organisations.

The initial versions of the Twin ID concept and the Digital Twin identifier registry along with their initial PoC implementations with URLs were described by [Aut2021b].

Trustworthiness can be achieved via Twin ID solutions on various levels. Trusting a plain URL or GS1 Digital Link requires that the DNS system itself and the holder of the domain are trustworthy. Additional trust can be established by signing URLs or documents with DIDs, although this requires that the user knows and trusts the signer. DIDs can also create decentralised chains of trust through the use of verifiable credentials. The chains can be used e.g. for delegating access management rights to a system through a chain of organisations.

Distributed ledger technologies can be used to provide immutable history for twin documents. This is done by hashing a twin document and storing the hash to a distributed ledger. By storing a hash, the contents of the twin document are not exposed publicly, but the existence of the hash at a certain time in the ledger means that the twin document existed at that point of time. To prevent tracking of unmodified twin documents across ledgers, a nonce (salt) is added to the twin document before hashing. A "low" degree of trustworthiness can be achieved by storing the hash to a fast ledger, whereas a high degree of trustworthiness can be achieved by further storing the hashes to a globally known secure ledger. A fast, privately hosted ledger can be cheap but only provides trustworthiness within a small community, whereas global ledgers such as Ethereum are expensive but provide a practically 100 % proof of the history. We can leverage an interledger solution to achieve a high level of trust while keeping the cost low as detailed in Section 3. It is important to however acknowledge the limitations of this solution, e.g. it is not possible to deduce the contents of the twin document from the ledgers, you can only verify that a certain document has existed at a certain point in time.

3.2.3 Semantic descriptions

Semantic descriptions are the contents of twin documents. The use of globally shared ontologies makes the twin documents machine-readable across implementations. This enables enhanced interoperability of real-world data across services.

For example, a visualisation software for city data can fetch the details of a data interface of a sensor device via a Semantic Twin, so that the user only needs to insert the Twin ID of the sensor to the visualisation software. Thanks to the semantic descriptions, the software will know the type of the sensor and visualise it in the correct way: a radar will be shown as a radar in the correct location and the observations of the radar will be included in the visualisation automatically.

A problem with using globally accepted ontologies for the semantic descriptions of twin documents in practice is that they do not cover enough use cases with high enough precision. Ontologies may also be difficult to find and many of them are not documented in a way that would enable fast adoption by people who are not deeply familiar with the conventions of the semantics field. In some cases, it may be necessary to create a new ontology, but the creation and publishing of them requires even more profound understanding of the conventions. We attempt to ease the barrier for adoption by introducing an ontology dedicated for Semantic Twins, described in the next section.

3.3 The Semantics of a Semantic Twin

One primary goal of the Semantic Twins is to enhance interoperability in the domain of Digital Twins. To achieve this, we utilise semantic technologies. Therefore, we'll introduce these briefly and discuss how IoT-NGIN utilises ontologies for semantic interoperability.

3.3.1 Basic Terms

A more comprehensive introduction into the topic of ontologies was already given in [D5.3]. Therefore, the introduction here will be kept shorter.

To avoid future misconceptions, it should be noted, that the two terms "Ontologies" and "Vocabularies" have the same meaning in the context of computer science, as, for example, the World Wide Web Consortium (W3C) states: *There is no clear division between what is referred to as "vocabularies" and "ontologies"* [Ont2022]. To answer the question, what ontologies are, we will start with the definition by the W3C: *Vocabularies define the concepts and relationships (also referred to as "terms") used to describe and represent an area of concern* [Ont2022]. This is compatible with other definitions [Brei2007] [Gua2009]. For example, the popular fiend-of-a-friend (FOAF) ontology about interpersonal relations contains terms for properties like "Name", "Gender" as well as relations like "knows" [Foaf2022].

As stated in the definition, ontologies focus on a single domain. However, ontologies can also include terms from other ontologies or allow relations to "foreign" concepts. For instance, the SAREF ontology [Saref2022] about IoT-devices utilises the W3C Geo ontology [Geo2022] for various kinds of location properties of IoT devices.

Ontologies focus on providing the vocabulary and the relations for a domain. They seldom incorporate information about individuals and instances of the classes. Thus, to gain value from the information of ontologies, this information, they need to be connected to actual data. This "fact oriented" result can then be called a knowledge base.

To allow the generation of such knowledge bases, e.g., as search engines do, it is advised to publish data with information about the related classes in an ontology. This practice is then referred to as "Linked Data".

The concept of linked data originates from the ideas of the semantic web, which is an effort to create a WWW-like web of machine-readable data. Humans can understand the semantics of information implicitly, but that is not the case for machines. Therefore, the data has to be annotated with semantic information, to allow algorithms to understand the data.

The inventor of the world-wide-web, Tim Berners-Lee has formulated the basic principles for linked data as follows [Ber2006]:

1. Use URIs for things
2. Use HTTP URIs
3. Make these HTTP URIs dereferenceable, returning useful information about the thing referred to
4. Include links to other URIs to allow discovery of more things.

The Semantic Twin concept aims to integrate into the semantic web by publishing the information about the twins as linked data. This way, algorithms can infer information about the twin and the real-world entity and new use-cases like the exchange of twins or the automatic aggregation of heterogeneous data are possible.

3.3.2 The Semantic Twin Ontology

To enable linked data for Semantic Twins, we have created an ontology on the domain of Semantic Twins. The main aspects of this domain are the Digital Twin, the real-world entity it describes and meta-information of the document. Especially for the first two aspects, various ontologies do exist. The Semantic Twin Ontology aims to incorporate these as well as possible instead of recreating an ontology for these domains, to keep the interoperability high and avoid the competing standards' problem.

We will mention a selection of relevant ontologies here: On the subject of IoT, the probably most relevant ones are the *Web of Things Thing Description* (WoT-TD) [WoT-TD], *oneM2M Base Ontology* [One2022], *SAREF* [Saref2022], and the *Semantic Sensor Network* (SSN) [Arm2017] and *Sensor, Observation, Sample, and Actuator* (SOSA) [SOSA] Ontologies. Wenbin et al. introduce more criteria to distinguish these ontologies and presented a more in-depth comparison in, we will focus on a short high-level update here [Wen2019]:

WoT-TD and SSN/SOSA are both recommendations from the World Wide Web Consortium (W3C). The former is built around the concept of a thing which has properties and interaction patterns, whereas SSN/SOSA represents sensors and actuators as well as observations and actuations. These are not part of WoT-TD.

The oneM2M Base ontology part of the global open standard oneM2M, pivoting around the concepts of Things, Devices, Services, Functions, Properties and more. A focus is put on machine-to-machine interactions rather than web applications.

SAREF is an ETSI standard supported by the European Commission, all pivoting around the concept of a Device. A main design element is the focus on easy extensibility, and ETSI themselves provide 12 extensions for SAREF. A mapping to the oneM2M ontology exists.

Depending on the aspect you are looking at, some of the previously mentioned Ontologies can also be used for the digital aspect of the twin. In addition, we briefly discuss other standards related to that aspect here as well. This is mostly based on the comparison done in [Jac2020]. The authors find the following standards which are part of some standardisation body or maintained by a big player in the industry and can be seen as an ontology in one or the other way:

- *Asset Administration Shell (AAS)* [AAS],
- *Digital Twin Definition Language (DTDL)* [DTDL],
- *Next Generation Service Interfaces-Linked Data API (NGSI-LD)* [NGSI-LD],
- *Open Data Protocol (OData)* [OData], and
- *SensorThings API (STA)* [STA]

AAS is mostly driven by the Platform Industries 4.0 network, where they do not use the term "Digital Twin" directly, but the concepts are the same. As the name already implies, the concept is pivoting around the aspect of an "asset", which is mostly the same as a Digital Twin. The ontology contains the aspects of resource description and resource discovery, whilst the standard for resource access has yet to be published. A ttl file is available [1].

DTDL is the ontology behind Microsoft's IoT and Digital Twin services. It only focuses on resource description and uses a custom type schema based on JSON-LD. Unfortunately, extending the ontology is not intended.

Another ontology originating from Microsoft is OData. It is intended to provide annotations for REST APIs, but the concepts are applicable to Digital Twins as well. OData is defining a custom language similar, but not equal, to JSON-LD. STA is strongly inspired by OData and adds some functionality like Message Queuing Telemetry Transport (MQTT) and geospatial

[1] https://github.com/admin-shell-io/aas-specs/blob/master/schemas/rdf/rdf-ontology.ttl

aspects. Like OData, it is difficult to integrate these in OWL-based ontologies, and thus they are just mentioned here briefly.

NGSI-LD is an ETSI standard for context information for IoT and Digital Twins. It contains building blocks to describe entities, relationships, and properties and also provides means for information exchange via a broker. NGSI-LD uses property graphs instead of RDF triplets.

With the overview of the related work in mind, we will continue by presenting the initial version of the IoT-NGIN Semantic Twin Ontology. By providing a machine understandable representation of the Semantic Twin solution, we can provide a semantic model for various kinds of Digital Twins and enhance interoperability. As many aspects of the solution have already been covered in various other ontologies, our solution is intended to provide the "glue" between these. The main classes forming the Semantic Twin Ontology can be seen in Figure 3.4.

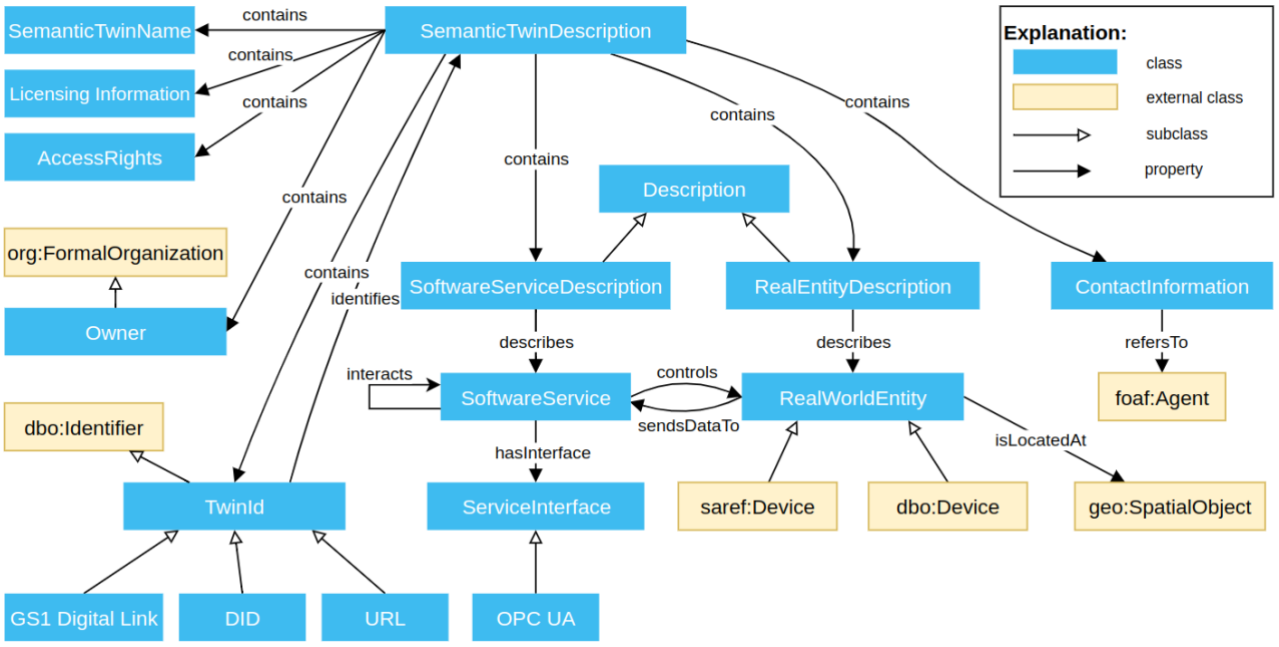

Figure 3.4 - Main Classes in the Semantic Twin Ontology.

Semantic Twins describe the essential information about a Digital Twin. Mainly the identity and owner of the Digital Twin, its real-world counterpart, access rights and terms of use, and relations to other Digital Twins. Therefore, we have a central *SemanticTwinDescription* class containing relations to the relevant classes covering the aspects from the Semantic Twin. The class *SoftwareServiceDescription* is describing the Digital Twin, whereas, *RealEntityDescription* is describing the physical counterpart. It can be observed that these have a described relation to a generic class, which can be set as superclass to classes coming from other ontologies, which are focussing on their special domain. Thus the Semantic Twin ontology can be used as a link between these ontologies.

Furthermore, the meta information of the Digital Twin do have their respective classes, here a focus on linking to standard ontologies was set. For example, the contact information uses the *friend-of-a-friend (FOAF)* ontology, which has widespread use in the semantic web, and TwinId is a subclass of *Identifier* from the DBpedia ontology.

To illustrate the usage of the ontology, we apply it to the example of a powertrain, which is also developed in IoT-NGIN. Please note, that this example does not resemble the real powertrain solution and is just here for illustrating the use of the Semantic Twin Ontology. In our example, we have a physical powertrain named "ABB Powertrain AP2000-1523" which has a Digital Twin in the "ABB Powertrain Control Online" software. The Powertrain is located at the "ABB Demonstration Site", has an Open Platform Communications United Architecture (OPC UA) interface, and "John Doe" is the responsible contact for this twin. The Semantic Twin describing all this is identified by the hosting url "http://twinbase.org/abb-powertrain".

Figure 3.5 shows the individual instances of the ontologies classes and their relations in this example.

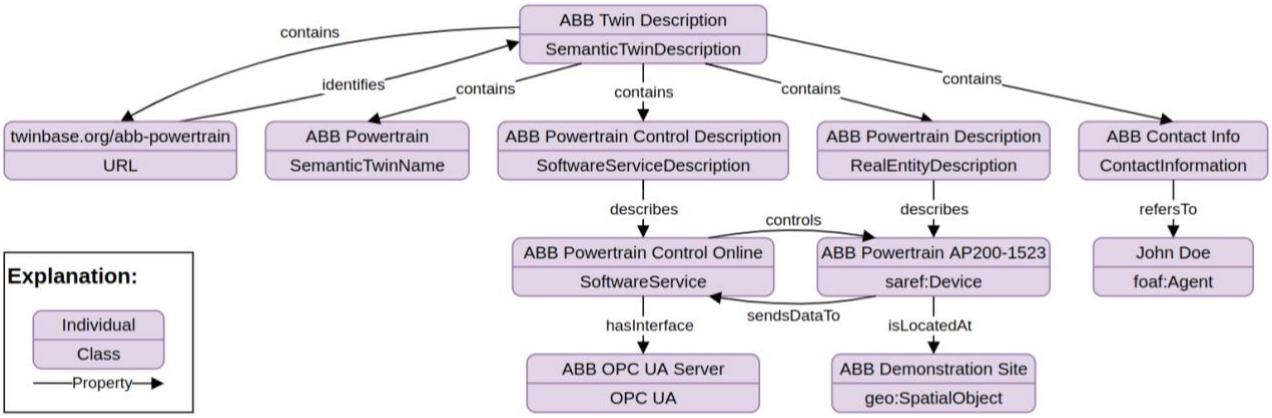

Figure 3.5 - Application of the ontology to a fictional powertrain example use case.

3.4 Next Steps

We now have the overall design of a Semantic Twin solution whose trustworthiness is enabled by SSI and DLT technologies. We have also experimented using parts of the solution in use cases. Next we will implement the solution more as a whole to use cases and perform an evaluation against the requirements laid out for the Semantic Twin solution in the previous deliverable [D5.3].

As an important individual result, we created an initial version of the Semantic Twin ontology. As next steps, we will apply this ontology to more examples and use cases from the IoT-NGIN project such as the Jätkäsaari Smart Junction from the Twin Cities Living Lab to validate the applicability. With a solid foundation, the integration into the Twinbase platform can be tackled, so that the hosted twin descriptions can be annotated semantically and machine understandability of the twins is enabled.

4 A Decentralised Interledger solution

This chapter presents the Decentralised Interledger Bridge (DIB) solution. First, the chapter summarises the need for multi-ledger transactions and how they can be met with a suitable interledger solution, the IoT-NGIN requirements for the interledger, and the existing interledger approaches. Based on the requirements, the Flexible Interledger Bridge (FIB) [Wu2021] developed in the EU Horizon 2020 project SOFIE [SOF2021] was then chosen as the basis for developing a decentralised solution, the *Decentralised Interledger Bridge (DIB)*. More details about available multi-ledger solutions and rationale for selecting the FIB as the basis of DIB can be found in IoT-NGIN deliverable D5.3 [D5.3]. The rest of the section then details the DIB solution.

4.1 Motivation for Interledger

Interledger technologies enable transactions that span two or more distributed ledgers. This section summarises why a separate technical solution is required for linking the ledgers, what benefits this approach enables, and what requirements a good interledger solution has to meet to be able to address the needs of IoT-NGIN.

4.1.1 Need for multi-ledger transactions

Distributed Ledger Technologies (DLTs) have been developed for over a decade, and they have been widely adopted due to the immutability and transparency provided by the decentralised secure storage, the distributed trust ensured by sophisticated consensus algorithms, and the automatic execution within the system enabled by features such as smart contracts [Zha2019]. According to their individual design goals, different DLTs have a varying emphasis, including the accessibility of data on the ledger (i.e., who is allowed to read or write on the ledger), the consensus mechanism adopted to reach agreement on ledger status, and the range of supported functionalities.

As DLTs have been deployed to more application areas, it has become clear that no single DLT is suitable for all use cases. Sometimes even the requirements of a single complex use case can easily exceed the strengths and capabilities of any single DLT. In such situations, combining multiple DLTs with different strengths and features can be a beneficial approach as it enables new functionality [But2016]. For instance, it might help improve the data integrity by utilising a highly trustworthy public ledger, while reducing the cost and latency of a system by keeping most of the heavy-lifting business logic in private ledgers.

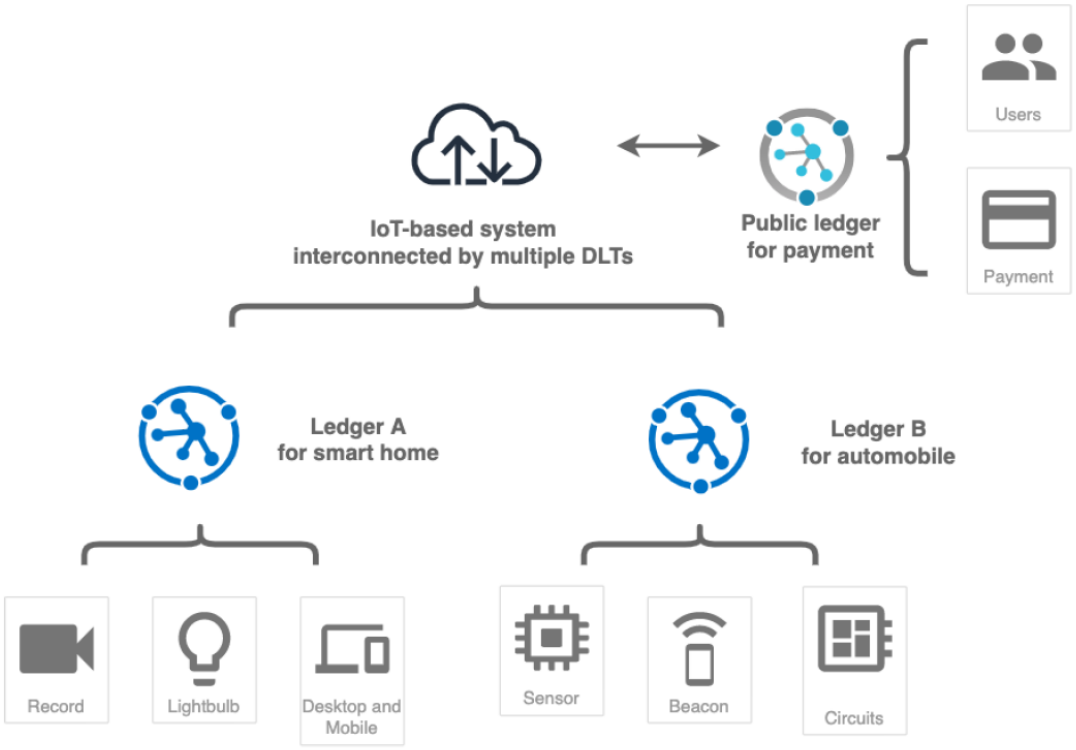

Figure 4.1 - An IoT-based system combining multiple DLTs.

A typical example are Internet of Things (IoT) systems, where an information sharing mechanism across multiple DLTs could help resolve the security, maintenance, and authentication issues in an automated manner [Has2019]. As illustrated in Figure 4.1, it is typical that IoT devices and services are connected to and backed by private distributed ledgers of individual vendors so that, e.g., the devices and equipment for a smart home interact with Ledger A, and the automobile sensors and circuits work together with ledger B. Then, a public ledger could be used for providing services for authentication and payment, and interlinking these three DLTs would enable a more complex (eco)system with additional functionality, e.g. payment services could be used with automobile ledgers at electricity charging stations.

4.1.2 Requirements of IoT-NGIN

The Interledger solution being developed (from here on: interledger) will be used in the IoT-NGIN project in several ways including the Smart Agriculture Living lab from WP7 (specifically the disease prediction and irrigation precision UC 3.1), the IoT intelligence empowered by federated machine learning from WP3, and also the Semantic Twins use case from WP5. Further, the IoT-NGIN architecture is expected to introduce many other uses for the interledger beyond the IoT-NGIN project itself.

Specifically, the Smart Agriculture Living Lab in WP7 could store state data related to disease findings and volume of irrigation water etc., while in the Smart Energy Grid Living Lab energy marketplace data needs to be stored. In WP3, trusted AI is targeted for federated machine learning: Zero Knowledge Proofs (ZKPs) of training datasets and trained federated machine learning models together with its parameters can be automatically stored on DLTs in form of

hash values and later utilised for verification by third parties to ensure they are not tampered with, while no actual data is released on the DLTs. Finally, Semantic Twins in WP5 utilise DLTs in a similar pattern, to ensure the integrity of relevant objects.

All the above use cases require auditability for logged data, but storing everything in a highly trustworthy public ledger would result in high costs and expose all data to potentially prying eyes. The low throughput of public ledgers can also become a problem in some cases. Storing everything in a private ledger would protect privacy, provide better throughput, and slash costs, but would also lack the high level of trust. A solution is to store the data in the private ledger and then leverage an interledger to automatically store a hash of the data at suitable intervals to the public ledger, this hash can also be salted by adding a random number to the calculation of the hash value to prevent guessing the data stored in the public ledger. This way, it is easy to verify whether the data in the private ledger has been tampered with while the overall costs are kept significantly lower as the usage of the expensive public ledger is reduced drastically.

Based on these different uses discussed above, 7 key requirements for the interledger solution can be identified as listed in Table 4.1 (table 2.1 in D5.3) and are detailed in the following text.

- REQ_IL_NF01: The interledger must be able to support the transfer of different types of data (so this excludes e.g. interledger solutions that focus exclusively on value transfers). Also, depending on the use, different types of DLTs may be utilised as part of the system, so the interledger solution has to be adaptable to different DLTs with relative ease.
- REQ_IL_NF02: The interledger must guarantee that the transactions across the ledgers are atomic, i.e. they happen completely on all the involved ledgers or not at all.
- REQ_IL_NF03: The interledger must provide transparency to the operations so that the correct operations of the interledger can be verified based on the data on the ledgers.
- REQ_IL_NF04: The interledger must operate so that non-repudiation for all parties of each individual transaction is guaranteed.
- REQ_IL_NF05: The interledger must be designed so that it can support a large number of transactions per second.
- REQ_IL_NF06: The interledger should minimise the overhead (cost, performance, storage etc.) for the application utilising the component for cross-ledger communication.
- REQ_IL_NF07: The interledger itself must support decentralisation, i.e. that the functionality is provided by a consortium of parties so that none of them can misbehave in any data transfer (e.g. change data payload, report invalid ledger transaction, or reject the transfer) without being detected by others. As a contrast, an interledger run by a single party has several limitations: the party has to be trusted by all users and it forms a single point of failure that can also pose problems for the resiliency and performance of the solution; a decentralised interledger helps address these limitations.

Table 4.1 - Requirements for the interledger solution [D5.3].

ID	Requirement	Description

REQ_IL_NF01	Generality	Must support general-purpose data transfers and be easily adaptable to different types of distributed ledgers.
REQ_IL_NF02	Atomicity	Must guarantee atomicity of transactions across the ledgers.
REQ_IL_NF03	Transparency	Must be transparent enough that the correct operation of all transactions can be verified based on the data on the ledgers.
REQ_IL_NF04	Non-repudiation	Must support non-repudiation so that the participants to a transaction cannot later deny their actions.
REQ_IL_NF05	Scalability	Must support a large number of transactions per second.
REQ_IL_NF06	Efficiency	Should keep the application overhead low
REQ_IL_NF07	Decentralisation	Must support decentralisation, where the interledger is run by a consortium of parties

4.2 Detailed description of the developed solution

This section describes the Decentralised Interledger Bridge (DIB) in more detail. The DIB implementation has been published as open-source[2].

Compared with its single node predecessor, the decentralised architecture of DIB design provides the shared trust among a consortium of participants for interledger transactions, while improving the robustness of the interledger data transfer via redundancy. To achieve a reasonable decentralised architecture for interledger, it is critical to make the following assumptions:

- Endpoints, which typically are smart contracts on distributed ledgers, at both source and destination of a data transfer will implement the interfaces required by DIB.
- Interledger nodes controlled by different parties in a consortium have the same full access to the endpoints, including both read and write operations.
- Interledger bridges at any nodes are equal in the sense that there is no special or admin bridge with superior functionality or access rights.

The high-level structure of the DIB design is illustrated in Figure 4.2 below. The architecture consists of a Decentralised State Management (DSM) layer in the centre for synchronising the common understanding of interledger data transfers (or interledger transaction interchangeably), and interledger nodes that host interledger bridge instances. In the

[2] https://gitlab.com/h2020-iot-ngin/enhancing_iot_cybersecurity_and_data_privacy/decentralised_interledger_bridge-dib.

illustration, endpoints (typically smart contracts on a distributed ledger) of each bridge are ignored for simplicity. The Connection Smart Contract (SCx in the figure) on DSM manages *unidirectional* interledger transfers between certain endpoints.

In this decentralised architecture, the interledger nodes should always have access to the DSM layer that is shared among the consortium of partners. The current implementation uses the Ethereum ledger for DSM due to wide availability of tools and ease of deployment. In addition to the Ethereum-based state manager, DIB also supports a *local state manager* which resides in the node's memory for cases where the extra resilience is not necessary and a single-node setup is sufficient. The local state manager also has higher performance than DSM as it does not have to synchronise the activities with other nodes.

While all the nodes have access to the DSM layer, only a single node should perform a transaction to the endpoint ledgers. Here DIB supports a timeout mechanism to provide extra resilience: if the node that is supposed to perform an endpoint transaction does not perform it within a certain amount of time, which is freely chosen by the deployer of the DSM, another node will take over this task.

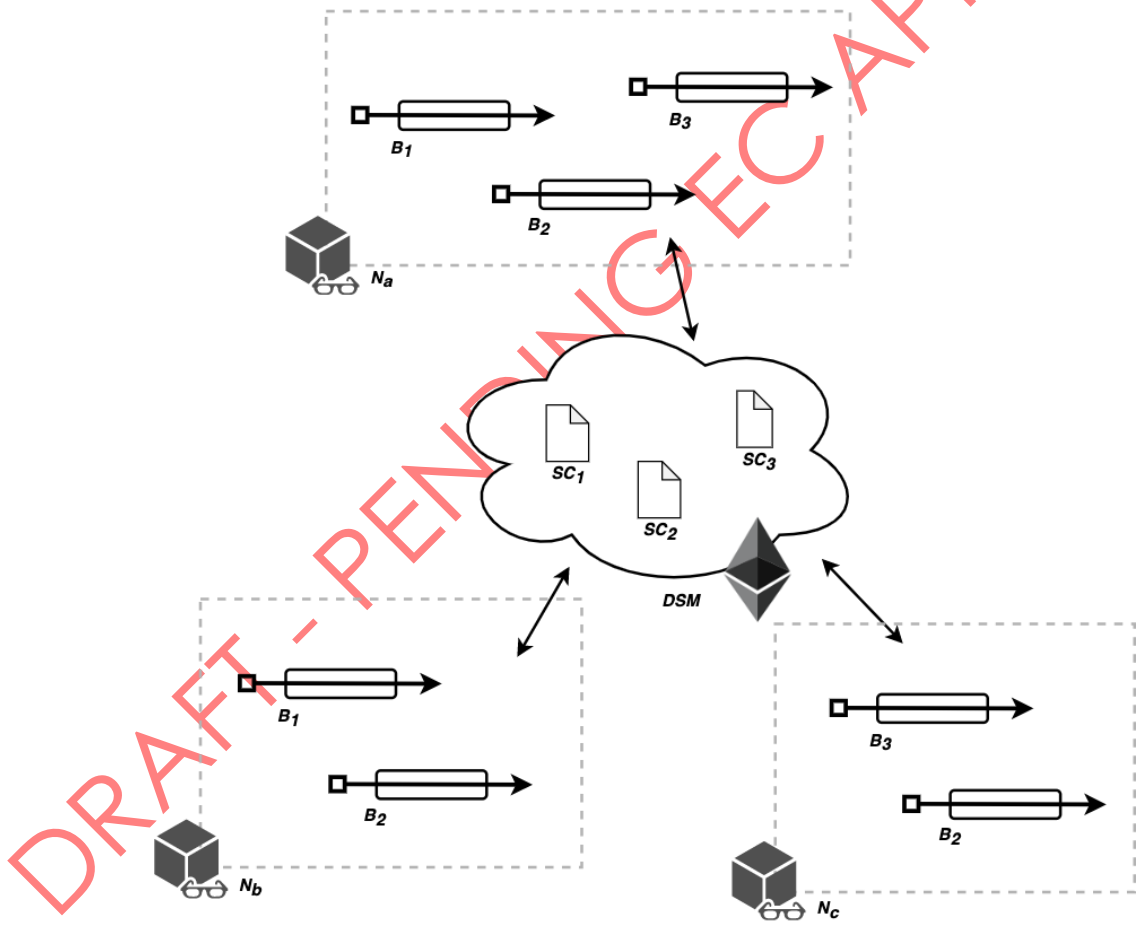

Figure 4.2 - DIB architecture consisting of nodes (Nx), bridge instances (Bx), and smart contracts (SCx).

4.2.1 Data flow of the Decentralised Interledger

A Decentralised Interledger transaction goes through a series of states from being initialised to finally committed, as shown in the following Figure 4.3. These states are recorded in DSM and will be explained in more detail on the following pages. Note that after the endorsement from other DSM participants, the state moves from Initialized / Accepted / Rejected / Committed / Aborted to the corresponding endorsed-state: InitializedEndorsed / AcceptedEndorsed / RejectedEndorsed / CommittedEndorsed / AbortedEndorsed states (not shown in the figure for simplicity).

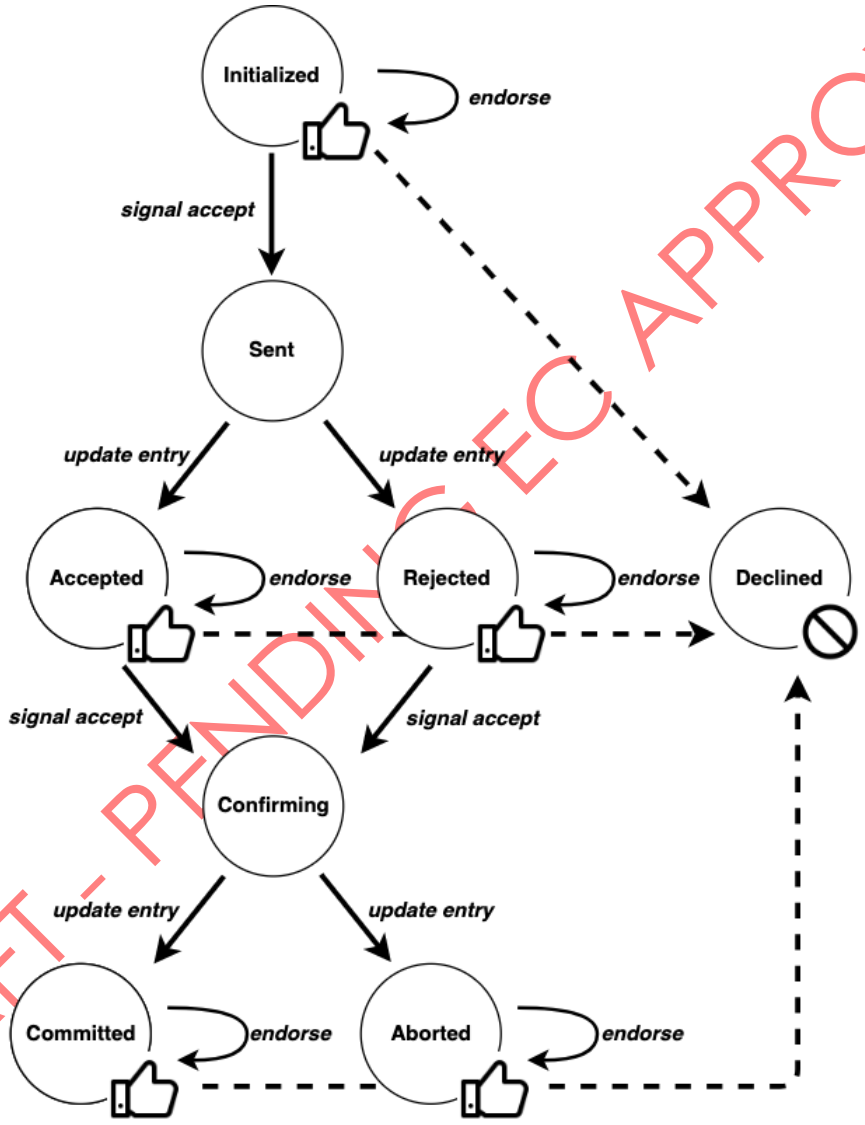

Figure 4.3 - Finite state machine of a decentralised interledger transfer.

The interledger transaction follows the flow illustrated in Figure 4.4, which consists of the following three major stages:

1. Receive a transaction from the endpoint Es from the source ledger
2. Send the transaction to the endpoint Ed at the destination ledger, and get back the response
3. Process and confirm transaction at Es again to conclude it

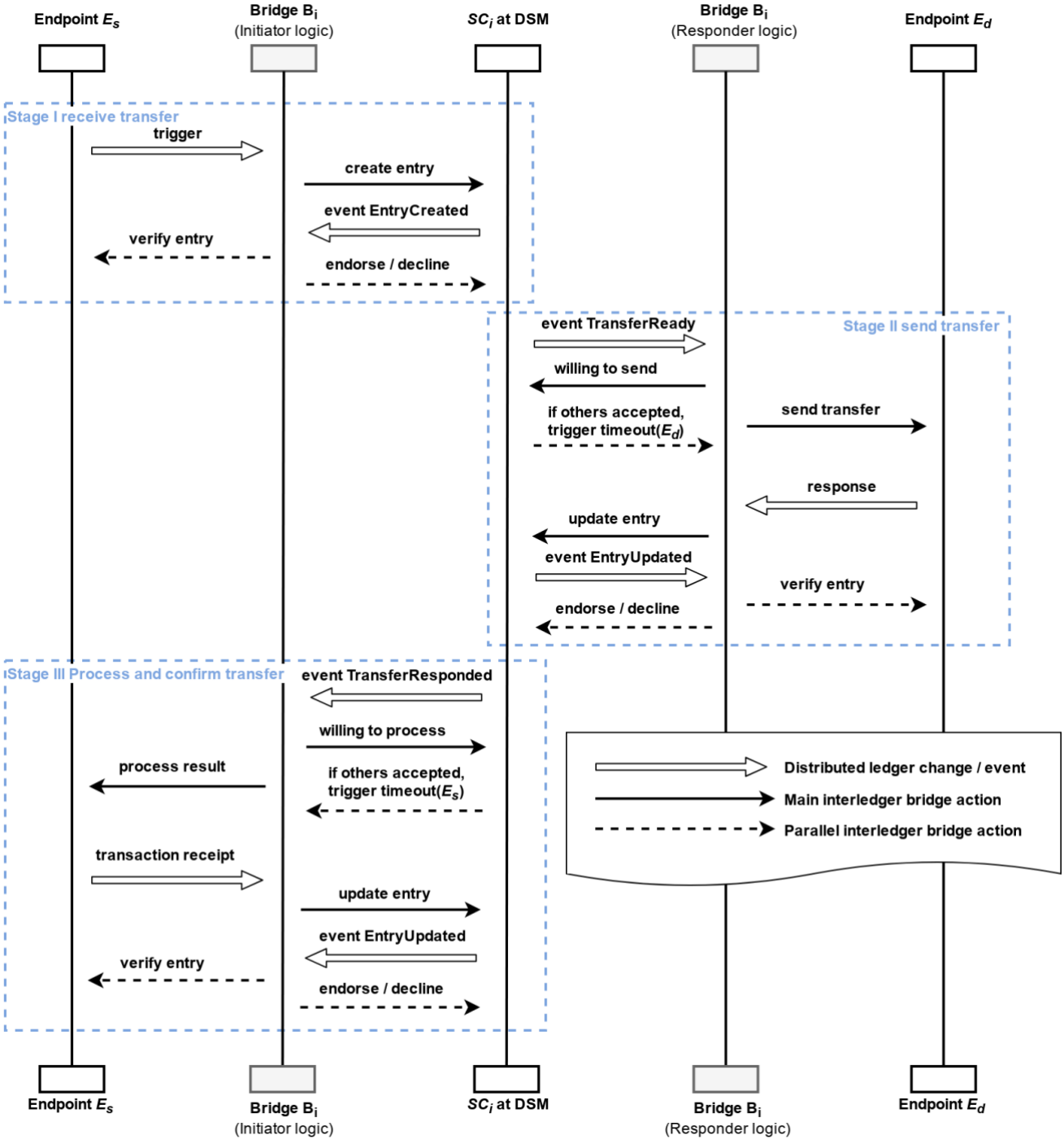

Figure 4.4 - Stages of Decentralised Interledger transaction.

All actions of the above stages will be recorded, updated, and audited by the other nodes §at the DSM layer in such a way that transparency is ensured. As a result, misbehaviour or malfunctioning of participants will be noticed by others. Meanwhile, at each step only one bridge instance will make the change to the endpoint, keeping the cost low and processing fast. Each stage has been described in detail below.

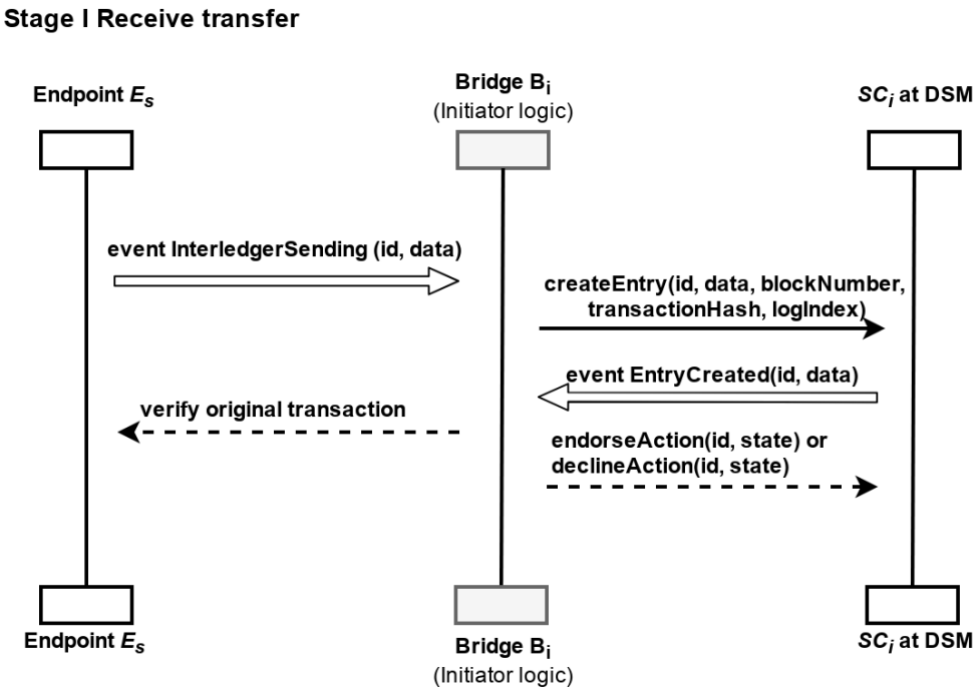

Figure 4.5 - Interledger transaction: Receiving transfer from the initiator.

This stage consists of the following steps:

1. All bridges will receive the *InterledgerSending(id, data)* event from the endpoint Es. Each of the bridges Bi will compete to create the transfer entry "t" at smart contract SCi at the DSM, via the *createEntry(id, data, blockNumber, transactionHash, logIndex)* method, here the last three parameters include the event details from the Es, which are necessary to validate the originating event.
2. The first successful createEntry transaction will trigger the event *EntryCreated(id, data)* to be emitted from the DSM ledger (the transaction will be in the Initialized state), after which all the participants will start checking its validity by verifying the original event on endpoint Es.
3. Based on the result of verification, the entry creation at the DSM gets endorsed or declined by all the participants via *endorseAction(id, state)* or *declineAction(id, state)* methods.

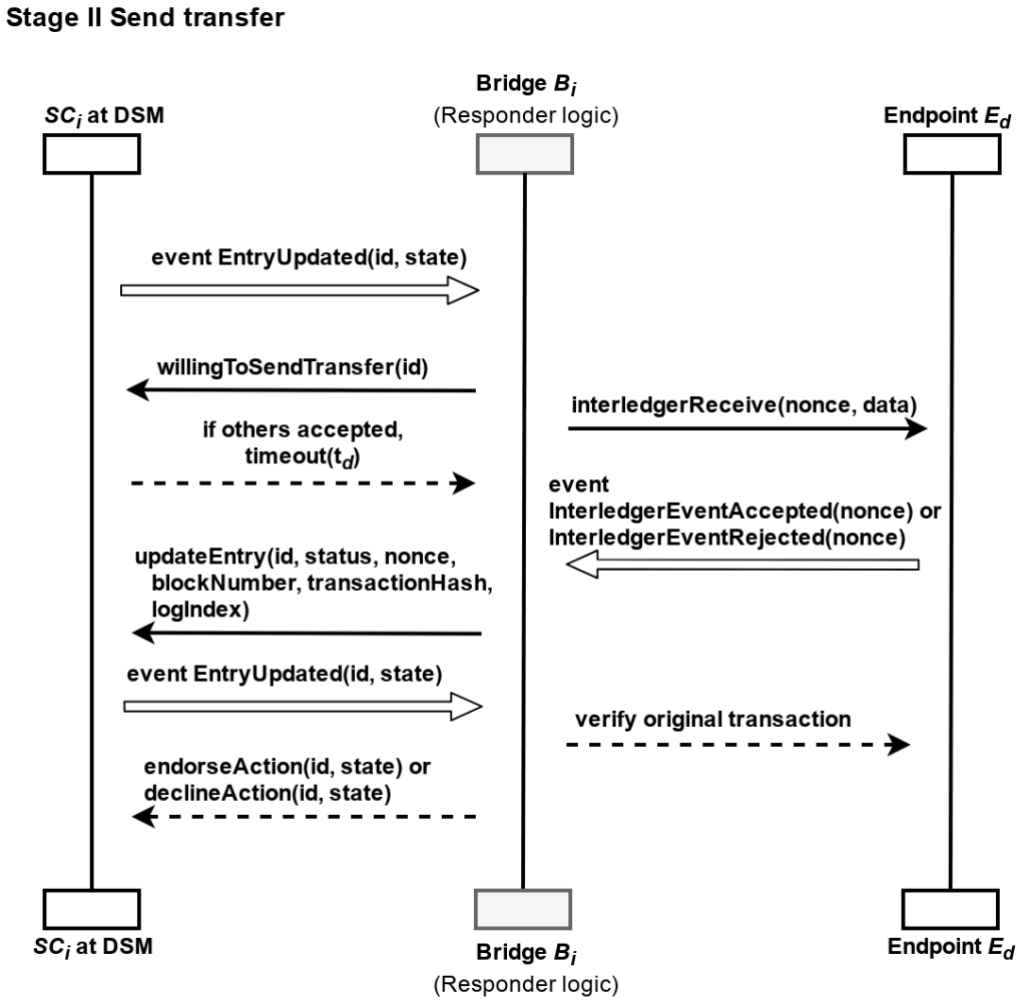

Figure 4.6 - Interledger transaction: Sending transfer to responder.

This stage consists of the following steps:

1. After the new transfer entry gets enough endorsements, which is by default the majority of participants but can be freely chosen, its internal state will change to InitializedEndorsed and the event *EntryUpdated(id, state)* is emitted from the DSM.
2. Each of the bridges Bi that received this event can signal the willingness to send the transfer to the endpoint Ed, via the *willingToSendTransfer(id)* method, which changes the transfer state to Sent.
3. The first successful bridge will send the transfer to the Ed via the *interledgerReceive(nonce, data)* method; other bridges then trigger the timeout logic, with reference time td. If the first successful bridge does not perform this transaction, the state will move to step 2. and another bridge will signal its willingness to send the transfer.
4. Once the application at Ed decides to accept or reject the transfer, based on the incoming data, the event *InterledgerEventAccepted(nonce)* or event *InterledgerEventRejected(nonce)* will be emitted, the sending bridge will update the transfer entry at the DSM accordingly using *updateEntry(id, status, nonce, blockNumber, transactionHash, logIndex)* method, which changes the transfer state either to Accepted or Rejected (depending on the application's response).
5. Corresponding event *EntryUpdated(id, state)* will be emitted from DSM.

6. All the bridges will check the validity of that update, by verifying the original transaction and related event on endpoint Ed.
7. Based on the result of verification, all the participants can endorse or decline the update at DSM via *endorseAction(id, state)* or *declineAction(id, state)* methods.

Note that signalling the willingness to send data here makes sure that only one bridge instance will make change to the connected distributed ledger. After the bridge actually makes the change on a ledger, all the bridges can then endorsed/declined the action.

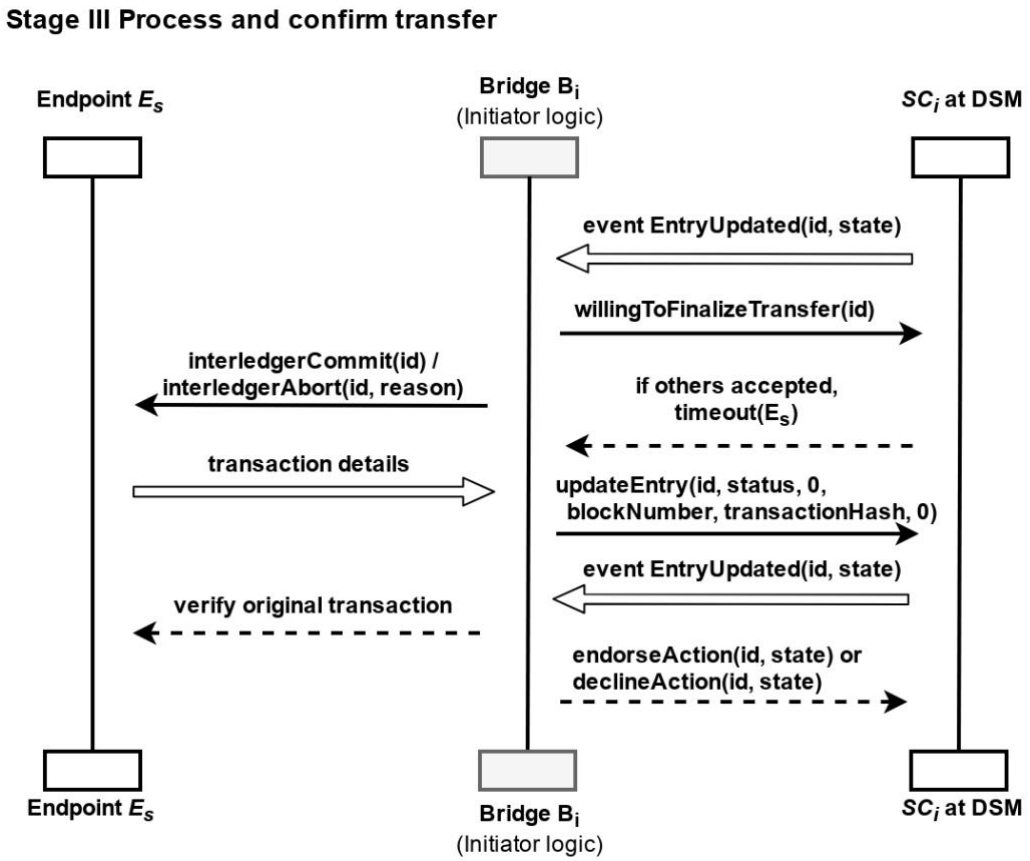

Figure 4.7 - Interledger transaction: Processing and confirming transfer to endpoint.

This stage consists of the following steps:

1. After the update of the previous stage gets enough endorsements, the transfer's internal state will change to AcceptedEndorsed or RejectedEndorsed and the event *EntryUpdated(id, state)* is emitted from the DSM.
2. Each of the bridges Bi that received this event can signal the willingness to finalise the transfer to the endpoint Es, via the *willingToFinalizeTransfer(id)* method, which changes the transfer state to Confirming.
3. The first successful bridge will finalise the transfer via the *interledgerCommit(id)* or *interledgerAbort(id, reason)* method; other bridges then trigger the timeout logic, with reference time ts.
4. After the transaction concludes, the confirming bridge will update the transfer entry at the DSM accordingly via *updateEntry(id, status, 0, blockNumber, transactionHash, 0)* method, which changes the transfer state either to Committed or Aborted.
5. Corresponding event *EntryUpdated(id, state)* will be emitted from DSM.

6. All the bridges will check the validity of that update, by verifying the original transaction on endpoint Es.
7. Based on the result of verification, the update at DSM gets endorsed or declined by all the participants via *endorseAction(id, state)* or *declineAction(id, state)* methods, which moves the transfer to CommittedEndorsed or AbortedEndorsed state.

If the transfer entry creation or update receives too many rejections at any point of time, the transfer moves to the Declined state.

4.2.2 Security properties of decentralised Interledger

The DIB provides decentralisation with the following benefits:

1. *Resiliency*. If one DIB node is not available to participate for any reason (node is down, lack of network connectivity, etc.), the interledger transactions will be successfully completed by other DIB nodes, as long as there is a sufficient number of nodes available. Even if one DIB node has already indicated its willingness to perform the transaction and then it is not able to do it, another node will take its place after the timeout.

2. *Auditability*. The DIB design allows multiple nodes (and parties) to join the DSM layer, which keeps track of interledger transactions. Therefore, all parties are able to verify that the transactions have been performed correctly.

However, the DIB can not prevent malicious node behaviour. Any node that has access to the source and destination ledgers can perform malicious transactions directly with these ledgers, bypassing the DIB. For example, the malicious node can signal to the source ledger that the transaction has been accepted/rejected immediately, or perform the *interledgerReceive()* transaction on the destination ledger with incorrect data or without the corresponding trigger from the source side. However, in these cases DIB still provides auditability, if all the nodes that have access to the source and destination ledgers participate in the DSM, then the malicious node can be identified by comparing transactions on the source, destination, and DSM ledgers.

By default a majority of nodes is sufficient to endorse/reject transactions, e.g. if there are 9 nodes in the DSM then endorsement from 5 of them is enough. This parameter can be freely chosen during the deployment of DSM smart contract, however changing it drastically may worsen the resiliency or security properties of DIB. E.g., if it is required that 90% of nodes endorse DSM transactions, then just having 11% of nodes offline or acting maliciously would stall the DIB process since there will not be enough nodes to endorse them.

4.3 Initial results and next steps

The DIB component satisfies all the requirements presented in Table 4.1:

- DIB supports transfer or any kind of data, instead of just monetary value. (REQ_IL_NF01)
- DIB provides atomic transactions, the transaction is confirmed/aborted on the Initiator ledger depending on the result of the Responder transaction. (REQ_IL_NF02)
- DIB provides transparency and non-repudiation since all of its actions are recorded to the ledgers. (REQ_IL_NF03 and REQ_IL_NF04)

- DIB component itself does not produce a high overhead and supports a large number of transactions. Performance and throughput of ledgers themselves is often the limiting factor. (REQ_IL_NF05)
- Ledger interfaces provided by the DIB component are simple and do not incur significant additional cost for the application smart contracts, running the component does not incur significant CPU overhead. (REQ_IL_NF06)
- DIB supports decentralisation as described in this section. (REQ_IL_NF07)

The following Table 4.2 presents initial test results of the following cases:

- Gametoken [Gam2022] transactions performed manually, without Interledger component
- Gametoken transactions using original single-node Interledger component
- Gametoken transactions using DIB and local state manager

All software components (ledgers, Interledger component, and test script) were run on the same computer. The throughput is relatively low since Ethereum ledger has not been optimised for a high throughput and a single transaction requires multiple ledger operations. Using Interledger component produces 26.5% reduction in TPS while, while using DIB with a local state manager lowers TPS by further 2.4%.

The DIB work is related to KPI 6.2: Supported cross-DLT Transactions per Second ≥ 10.000. As an almost unlimited number of DIBs can be run in parallel, this number is reachable with sufficiently many DIBs. Detailed Evaluation of this KPI will be reported in D5.5. There are also plans to test the DIB performance with nodes running in different countries.

Table 4.2 - Initial Performance Results of DIB.

Case	**Throughput (transactions per second, TPS)**
No Interledger	11.7
Single-node Interledger	8.6
DIB with local state manager	8.4
Test setup	**Hardware:** Ryzen 7 Pro 4750U (8-core 1.7-4.1GHz) mobile CPU **Software:** web3.py 5.28 Geth 1.10.23-stable using IPC sockets

5 Self-Sovereign Identity Technologies

The use of SSI technologies for the triplet's identity and trustworthiness has already been discussed in Section 3. This chapter, therefore, provides details of the two other uses for the SSI technologies explored in IoT-NGIN, i.e. Verifiable Credential based decentralised on-device access control with constrained IoT Devices and QR code and GS1 Digital Link based discovery mechanisms for the Triplet.

5.1 Verifiable Credential-based Access Control on Constrained IoT Devices

Verifiable Credentials allow flexible and privacy-preserving access control solutions. E.g., suppose there is a factory that has outsourced the maintenance to a separate company. The technician working for the maintenance company needs to receive temporary access to factory premises and to certain machines there, but the factory does not need to learn about the technician's real identity or whether the technician is the same as the one who visited the factory previously.

This subsection describes a verifiable credential-based access control solution that can be used directly on the constrained devices, i.e. the constrained device such as ESP32 microcontroller verifies the credentials and enforces the access control policies. The solution is also available as open source [3].

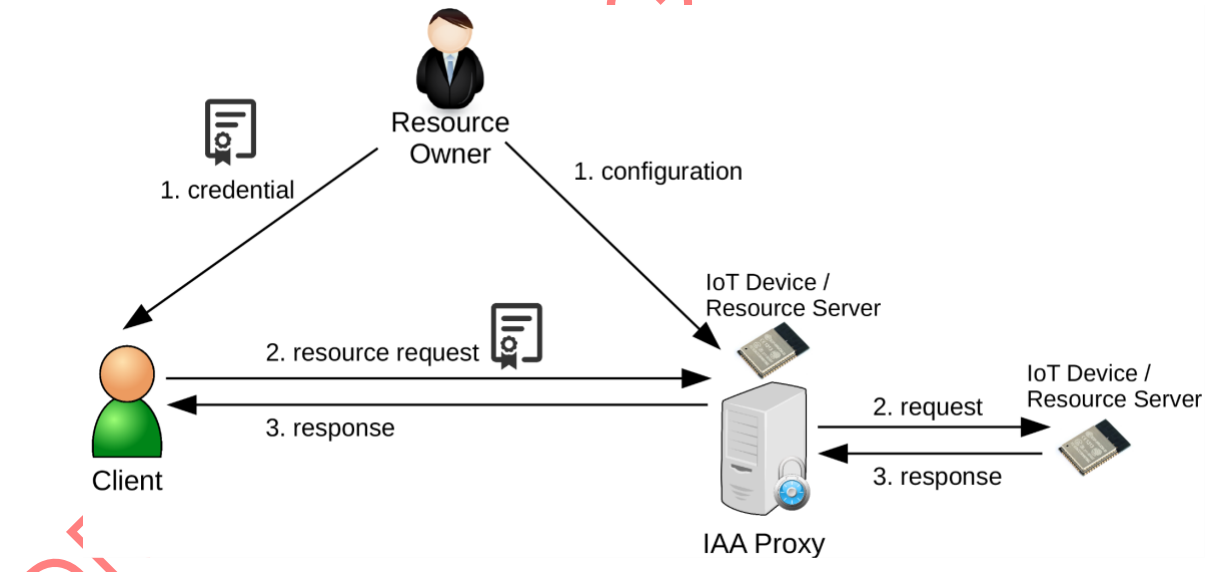

Figure 5.1 - Overview of the SSI Access Control component.

[3] https://gitlab.com/h2020-iot-ngin/enhancing_iot_cybersecurity_and_data_privacy/privacy-preserving-self-sovereign-identities

Figure 5.1 provides an overview of how the SSI component can be used to grant and verify access to the Resource Server, which can be for example an IoT device. The Resource Owner and Client are identified using Decentralised Identifiers (DIDs). In the first step, the Owner configures the *Identity, Authentication and Authorisation* (IAA) proxy and grants a Verifiable Credential (VC) to the Client, which denotes that the Client has a right to access some Resource. The Client uses this credential to contact the IAA proxy or the actual IoT device, which will then verify the credential and grant a read or write access to the resource. In a case of the IAA proxy, it will forward the request to the actual Resource Server, which does not need to understand SSI technologies or even handle the cryptographic operations.

In more detail, the credential is encoded as a standard JSON Web Token (JWT) and in order to prevent replay attacks, the client also constructs a Demonstrating of Proof-of-Possession (DPoP) proof when accessing the resource. Both the credential and the DPoP proof will be verified by the IAA proxy or the actual device.

The SSI component provides the following functionality:

- Tools for identity and key management, including the creation of credentials encoded as JWTs and DPoP proofs. For DID methods, did:self and did:key are supported and the Ed25519 EdDSA signature scheme is supported for cryptographic signatures.
- IAA proxy and simple resource server based on existing py-verifier work [4].
- Verifier for ESP32-based embedded devices, which allows full verification of access control credentials to be performed on an embedded device.

The performance on the constrained device is good, the full JWT + DPoP verification consisting of two signature verifications takes just 160ms on the low cost ESP32 device. Therefore, the whole process of accessing the protected resource takes well below one second, which is a sufficient performance from user experience point of view [Fot2022].

5.2 Triplet discovery using QR codes and GS1 Digital Links

A GS1 Digital Link [5] converts a barcode, either one or bi-dimensional, into a web address that contains the information on a product the barcode refers to. GS1 digital links are used to discover the locations of the Digital and Semantic Twin of an entity Triplet.

The discovery protocol begins with a user in front of a barcode, e.g. a QR code, attached to a real-world entity, such as an IoT device, and is shown in Figure 5.2.

[4] https://github.com/mmlab-aueb/py-verifier

[5] https://github.com/gs1/GS1_DigitalLink_Resolver_CE

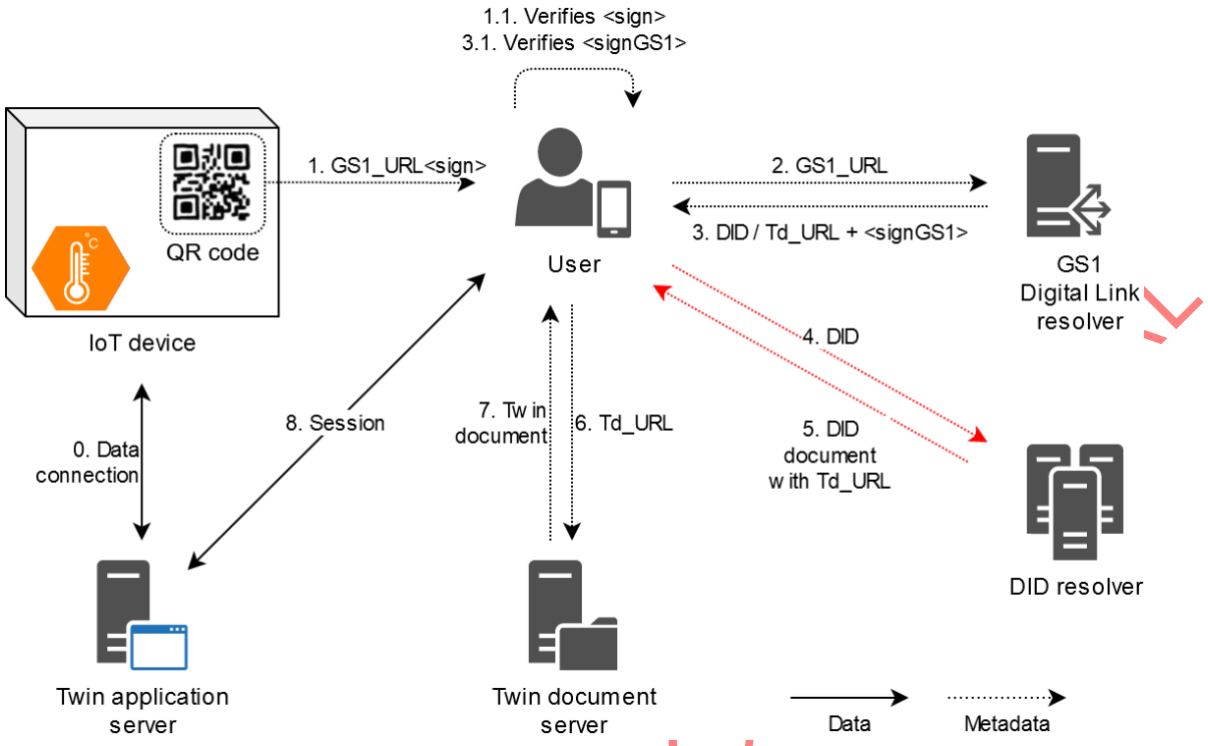

Figure 5.2 - The triplet discovery protocol.

The QR code encodes the URL (like https://gs1resolver.iot-ngin.eu/gtin:123456 which has not yet implemented) and the GTIN number of the device to the GS1 Digital Link Resolver server. The User scans the QR code with a smartphone using a dedicated app that queries the GS1 Digital Link Resolver Server to get either the locations or the DIDs of the Digital and Semantic Twins. This differentiation depends on the DID method used by the entity triplet:

- If the DID method is a ledger-based one that allows adding information to a DID into the ledger, such as *did:ethr*, the GS1 Digital Link Resolver server returns the DIDs of the Digital and Semantic Twins whose resolution, shown in red arrows in Figure 5.2, gives the User their DID documents containing the location parameters;
- Otherwise, if it is not possible to add data to DIDs, such as in *did:key* DID method, the Resolver server returns the User the location of the Digital and Semantic Twins.

The twin description document describes the available data interfaces in a structured way, possibly including semantic information as well. For example, the drive unit gathers measurements of the same physical quantities from all powertrains, e.g. motor speed, torque, and current. However, the underlying protocols and data structures used to collect this data may vary. The twin description allows the application programmer to handle all powertrains in a consistent and structured manner. For example, the addition of new powertrains, or changes to existing powertrain implementations, can be handled with less effort, as the overall structure of the twin description remains the same. This becomes more apparent in larger and more complex use cases, which may consist of multiple parties and hundreds of devices.

In either case, the user accesses the digital or the Semantic Twin. The figure shows the user accessing the Semantic Twin and getting the Twin Document that allows them to open a

session with a Twin Application Server and perform operations (depending on their level of privilege).

To guarantee the QR code the User is scanning is the original one, and it has not been switched with a malicious one, the QR code could embed the digital signature of the organisation that issued it. This is feasible since a QR code can encode up to 3 KB of data. Before accessing the URL (like https://gs1resolver.iot-ngin.eu/gtin:123456&<digital signature>, which has not yet implemented) encoded in the QR code, the user's app verifies the signature with the organisation's public key (step 1.1 in the figure). Similarly, the data returned by the GS1 Digital Link Resolver server is digitally signed and verified by the User (step 3.1 in the figure).

The code of the GS1 Digital Link Resolver server can be found at [6].

6 https://gitlab.com/h2020-iot-ngin/enhancing_iot_cybersecurity_and_data_privacy/qr-discovery

6 Integrating the solutions

The solutions previously described are used together to enable data sovereignty by making IoT data and services accessible in a trusted, auditable, and controlled way. In particular, to support the installation, configuration, and maintenance of the Digital Twin and the Semantic Twin of IoT devices following the SSI paradigm while protecting the privacy of the users interacting with the Twins.

Section 6.1 describes a demo for the configuration of IoT devices to showcase how the integration of the solutions works. Section 6.2 presents the Living Lab use cases that adopt, implement, and validate such integration.

6.1 IoT devices configuration demo

This demo integrates all the above technologies to demonstrate how to easily discover, protect, and configure the IoT Triplet while protecting the privacy of the individual users and providing good user experience through low-latency validation. The key actors of the demo use case are illustrated in Figure 6.1.

The *Traffic Department* of a City buys *IoT Devices* from a *Manufacturer* and wants to install them to a Smart City project. The Traffic Department initialises a device and its Digital Twin and Semantic Twin with the basic information required to delegate the setup to an external *Installer Company*. Moreover, the Traffic Department creates a QR code for each device, embedding a GTIN number that, once resolved by a GS1 Digital Link Resolver server, provides the locations to access the device's Digital Twin and Semantic Twin.

The Installer Company employs one or more *Installers Employees* to go around the city and install the devices (and possibly maintain them afterwards). To finalise the installation of a device, an Installer Employee accesses the Digital Twin and the Semantic Twin of that device. To access them, they require a credential that can be obtained from the Installer Company

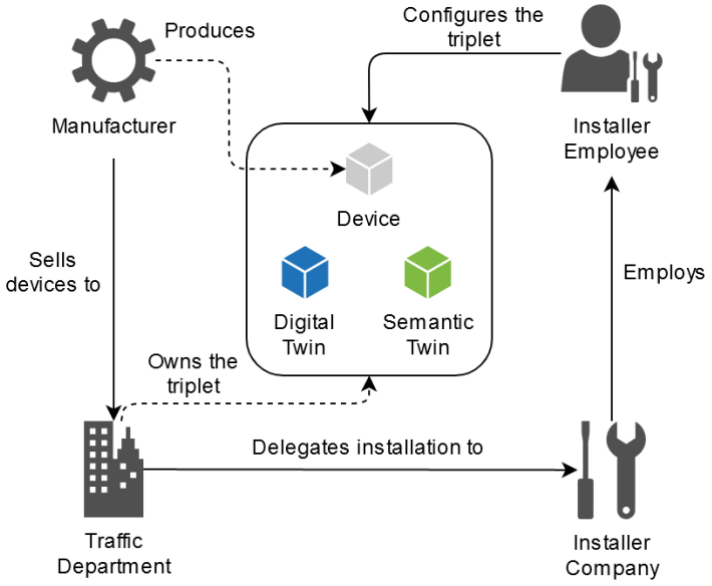

Figure 6.1 - Illustration of the demo.

Authorization Server. The Employee scans the QR code on the device with a mobile phone application. Once scanned, the QR code redirects the Employee to the Semantic Twin. At access request, the server hosting the Semantic Twin, e.g. the Twin document server shown in Figure 5.2, begins an access control protocol to know the privileges of the person who is requesting the access. With the QR code being accessible to anyone, any citizen of the City could potentially get access to the Semantic Twin to view information about the Device. This could be a wanted feature of the Smart City project.

With this demo, we aim to address the following problems:

- Discovering the Twins related to an IoT Device;
- Enabling secure access to the triplet;
- Trusting the data received by the triplet;
- Protecting people's privacy;
- Detecting malicious activities on the triplet.

6.1.1 Demo Description

The system resulting from the integration of the solutions described in this document is structured as follows.

The Traffic Department, who owns the entity triplets, is responsible for setting up the entries for each triplet in the GS1 Digital Link Resolver server and printing the correspondent QR codes. Moreover, the Traffic Department issues a VC_{Dept} to the Installer Company to configure the triplets: this VC has a "delegate" option so that the Installer Company can delegate the installation rights to its Employees.

The Installer Company sets up an Authorization Server that, being delegated by the Traffic Department, issues a VC_{Config}, alongside VC_{Dept}, to the Employees to configure the triplets.

Following the SSI approach, any actor issues or receives VCs from or to their DIDs. Figure 6.2 shows the DIDs paired to each actor in this demo. Actors such as the Traffic Department, the GS1 Digital Link Resolver server, and the Installer Company Authorization server may need to attach additional information to their DIDs. Thus they could use a ledger-based DID method. Instead, others may only need DIDs as pseudonyms, therefore a non-ledger-based DID method would be suitable.

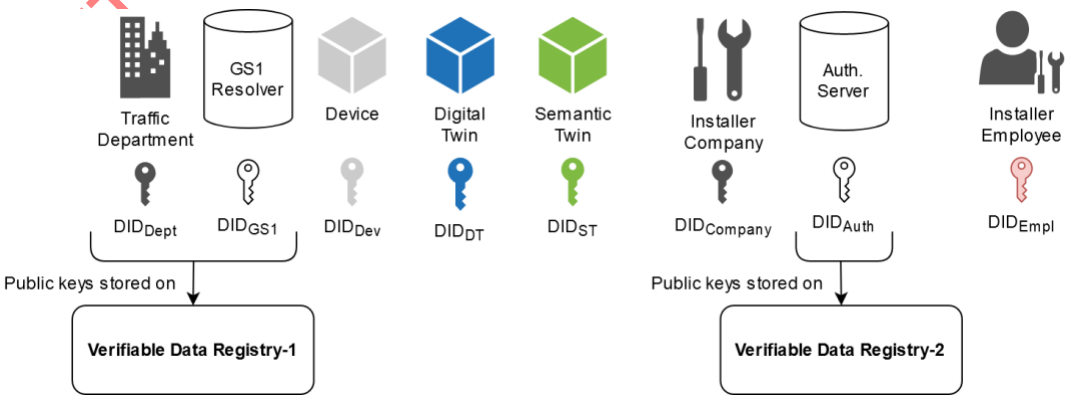

Figure 6.2 - Illustration of the DIDs used by the actors.

Before configuring the triplets, the Installer Employee generates their DID_{Empl} and requests the Authorization Server a VC_{Config} to be issued, alongside VC_{Dept}, to the newly generated DID. Examples of attributes, or claims, of VC_{Config} are the type of Devices the Employee will configure, their location, and the duration of the validity of the credential (e.g. 24 hours). The Installer Company and the Employee need to agree on a common secret parameter or a similar solution to ensure only an Employee of the Installer is able to request such credentials.

When the Employee reaches a Device, scanning the QR code triggers the discovery protocol described in Section 5.2.

When the Employee locates the server hosting the Semantic Twin, they can access it following the access control protocol described in Section 5.1. In particular, the Employee signs a DPoP with their DID_{Empl} and sends it alongside the credentials VC_{Dept} and VC_{Config} to an IAA proxy to the Semantic Twin for access control. The IAA proxy checks:

- The validity period of VC_{Config};
- The attributes in VC_{Config} match its attributes (and the Employee is not accessing to the wrong device);
- The signature in VC_{Dept} is verified by DID_{Dept};
- The signature in VC_{Config} is verified by DID_{Auth};
- Ensure DID_{Auth} is delegated by VC_{Dept};
- The signature in DPoP is verified by DID_{Empl}.

If all checks are successful, the proxy allows access to the Semantic Twin to retrieve the Twin Document. To make the Twin Document data more trustworthy, the Semantic Twin can sign it with its DID_{ST}. Moreover, the security, integrity, and accessibility of the Twin document is helped by integrating DLTs and the Interledger component, whose functionality is described in Section 4. The protocol is shown in Figure 6.3.

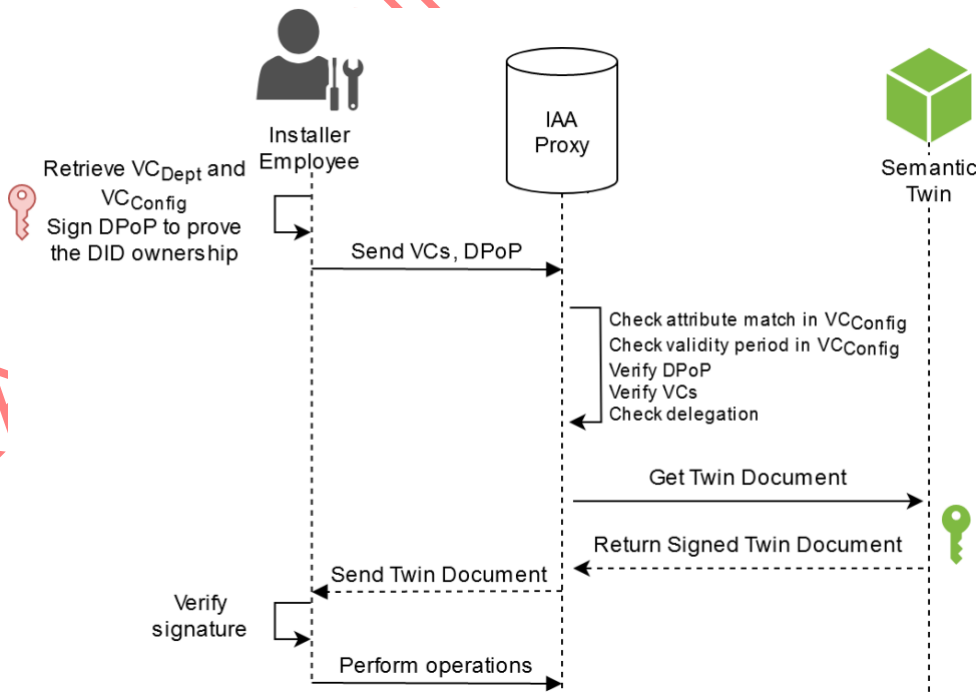

Figure 6.3 - The access control protocol to the Semantic Twin.

The problems mentioned in this section are addressed as follows:

- Discovering the Twins related to an IoT Device: this is solved by the GS1 Digital Link Resolver server;
- Enabling secure access at the triplet: this is solved by the VCs issued by the actors the access control protocol executed by the IAA proxy or on-device validation;
- Trustworthiness of the data received by the triplet: this is solved by applying digital signatures to the QR code, to the response data returned by the GS1 Digital Link Resolver server, and to the data returned by the Semantic Twin;
- Protecting people's privacy: this is solved by hiding the identity of the Installer Employee during access time to the Semantic Twin behind an ephemeral DID;
- Accountability for malicious activities on the triplet: if a malicious behaviour is detected on a Semantic Twin, the Installer Company can link the DID used to access the Semantic Twin to the identity of the Employee who used that DID to request the VC used to access to the Semantic Twin, and take action.

6.2 Living Lab use cases

This section presents the Living Lab use cases (UCs) that integrate the solutions presented in this document. Table 6.1 shows the integration of the technologies within the use cases. As shown in the table, 6 use cases out of 10 need at least two of the technologies that are presented in this deliverable, in particular SSI technologies, thus motivating their importance and enabling extensive validation of the solutions. In particular, it is worth noticing that the use cases where such technologies are more relevant are in the field of Smart Cities, Smart Energy, and Industry 4.0. A detailed description of each UC can be found in D7.2.

Table 6.1 - Use cases integrating the solutions.

IoT-NGIN Technology	Smart Cities			Smart Agriculture		Industry 4.0			Smart Energy	
	UC1	UC2	UC3	UC4	UC5	UC6	UC7	UC8	UC9	UC10
WP5 - Enhancing IoT Cybersecurity & Data Privacy										
Decentralised Interledger Bridge	✓	✓						✓		✓
Privacy Preserving Self-Sovereign Identities (SSIs)	✓	✓	✓						✓	✓
Semantic Twins	✓	✓	✓					✓	✓	

7 Conclusions

This document discusses the technical solutions from Tasks T5.3-5 including Semantic Twins, ontologies, multi-ledger transactions, and Self-Sovereign Identities that can be utilised to tackle the problems in the domain of IoT systems.

Based upon various needs in use cases within the IoT-NGIN project, technical solutions for each area were planned and successfully developed. With them, the developed technologies can successfully be deployed to achieve the goals of the work package.

The design of the DLT-enabled Semantic Twin solution was presented in this deliverable, in line with KPI T6.4. Parts of the solution have been implemented: trustworthiness enabled by SSI and DLT technologies has been experimented and an initial version of Semantic Twin ontology has been created. The comprehensive Semantic Twin solution will be applied to IoT-NGIN use cases and presented in Deliverable 5.5.

Decentralised Interledger Bridge (DIB) has been implemented to allow transfer of information between distributed ledgers. Due to decentralisation, DIB is resilient in case of node failures.

SSI technologies are also used for decentralised on-device access control with constrained IoT Devices and QR code and GS1 Digital Link based discovery mechanisms.

This document describes a demo for the configuration of IoT devices to showcase how the integration of the mentioned solutions works. Totally 6 use cases out of 10 need at least two of the technologies that are presented in this deliverable, in particular SSI technologies, thus motivating their importance and enabling extensive validation of the solutions.

The final versions of the solutions, their deployment in the Living Labs and the validation of the solutions in the Living Labs will be described in the upcoming deliverable 5.5.

8 References

[AAS] Asset Administration Shell Specifications, Plattform Industrie 4.0. 23/11/2020
https://www.plattform-i40.de/IP/Redaktion/EN/Standardartikel/specification-administrationshell.html

[Ala2021] R. Ala-Laurinaho, "API-based Digital Twins - Architecture for Building Modular Digital Twins Following Microservices Architectural Style," Doctoral dissertation, Aalto University, 2021.
http://urn.fi/URN:ISBN:978-952-64-0594-0

[Arm2017] Armin Haller et al. Semantic Sensor Network Ontology. Technical Specification
OGC 16-079. W3C & OGC, Oct. 17, 2017. url:
https://www.w3.org/TR/vocab-ssn/

[Aut2021a] J. Autiosalo, "Discovering the Digital Twin Web - From singular applications to a scalable network," Doctoral dissertation, Aalto University, 2021.
http://urn.fi/URN:ISBN:978-952-64-0621-3

[Aut2021b] J. Autiosalo, J. Siegel, K. Tammi "Twinbase: Open-Source Server Software for the Digital Twin Web", *IEEE Access*, vol 9, pp. 140779-140798, 2021.
https://doi.org/10.1109/ACCESS.2021.3119487

[Ber2006] T. Berners-Lee "Linked Data" https://www.w3.org/DesignIssues/LinkedData

[Brei2007] K. Breitman, M.A. Casanova, and W. Truszkowski, Semantic Web: Concepts, Technologies and Applications, 2007.

[But2016] V. Buterin, "Chain interoperability," R3 Research Paper, 2016.

[DTDL] Digital Twins Definition Language
https://github.com/Azure/opendigitaltwins-dtdl

[D1.1] IoT-NGIN D1.1 - Definition analysis of use cases and GDPR Compliance
https://iot-ngin.eu/index.php/deliverable/

[D1.2] IoT-NGIN D1.2 - IoT meta-architecture, components and benchmarking
https://iot-ngin.eu/index.php/deliverable/

[D5.3] IoT-NGIN D5.3 - Enhancing IoT Data Privacy & Trust

https://iot-ngin.eu/index.php/deliverable/

[D6.2] IoT-NGIN D5.3 - Integrated IoT-NGIN platform & laboratory testing results
https://iot-ngin.eu/index.php/deliverable/

[D7.3] IoT-NGIN D7.3 - Living Labs use cases intermediate results
https://iot-ngin.eu/index.php/deliverable/

[Foa2022] Foaf Vocabulary Specification
http://xmlns.com/foaf/0.1/

[Fot2022] N. Fotiou, et al. "Capabilities-based access control for IoT devices using Verifiable Credentials." 2022 IEEE Security and Privacy Workshops (SPW). IEEE, 2022.
https://doi.org/10.1109/SPW54247.2022.9833873

[Geo2022] Basic Geo (WGS84 lat/long) Vocabulary
https://www.w3.org/2003/01/geo/

[Gua2009] Nicola Guarino, Daniel Eberle, and Steffen Staab. Chapter: "What is an ontology?" in Handbook on ontologies pp. 1-17, 2009.

[Has2019] M. U. Hassan, M. H. Rehmani, and J. Chen, "Privacy preservation in blockchain based IoT systems: Integration issues, prospects, challenges, and future research directions," Future Generation Computer Systems, vol. 97, pp. 512–529, 2019.

[Jac2020] M. Jacoby, and T. Usländer, "Digital Twin and Internet of Things—Current Standards Landscape," *Applied Sciences* 2020, *10*, 6519.
https://doi.org/10.3390/app10186519

[Mat2022] J. Mattila, R. Ala-Laurinaho, J. Autiosalo, P. Salminen, and K. Tammi, "Using Digital Twin Documents to Control a Smart Factory: Simulation Approach with ROS, Gazebo, and Twinbase," *Machines* 2022, 10(4), 225.
https://doi.org/10.3390/machines10040225

[NGSI-LD] Duncan et al., "NGSI-LD API: for Context Information Management ," ETSI White Paper No. 31, 1st ed., 2019.I
https://www.etsi.org/images/files/ETSIWhitePapers/etsi_wp31_NGSI_API.pdf

[Odata] Open Data Protocol
https://www.odata.org/

[One2022] Ontologies used for oneM2M
https://www.onem2m.org/technical/onem2m-ontologies

[Ont2022] Vocabularies for Ontologies
https://www.w3.org/standards/semanticweb/ontology

[Saref2022] Smart Applications REFerence Ontology, and extensions
https://saref.etsi.org/

[SOF2021] SOFIE project: Secure Open Federation for Internet Everywhere," https://www.sofie-iot.eu/, (Accessed on 03/12/2021).

[SOSA] SOSA Ontology
https://www.w3.org/2015/spatial/wiki/SOSA_Ontology

[STA] OGC SensorThings API
https://www.ogc.org/standards/sensorthings

[Twinbase] Twinbase: Server software for hosting digital twin documents
https://github.com/twinbase/twinbase

[Wen2019] Wenbin lin et al. Review of Standard Ontologies for the Web of Things

[WoT-TD] Web of Things (WoT) Thing Description. W3C Recommendation 9 April 2020
https://www.w3.org/TR/wot-thing-description/

[Wu2021] Wu, L., Kortesniemi, Y., Lagutin, D., & Pahlevan, M. (2021, September). The Flexible Interledger Bridge Design. In 2021 3rd Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS) (pp. 69-72). IEEE.

[Zha2019] J. Zhang, S. Zhong, J. Wang, L. Wang, Y. Yang, B. Wei, and G. Zhou, "A review on blockchain-based systems and applications," in International Conference on Internet of Vehicles. Springer, 2019, pp. 237–249.

9 Annex 1: Powertrain use case

The use case UC8 takes place at ABB's premises in Helsinki. Two factory sites have up to 6 powertrains with varying sizes of motors. The powertrain is used to describe the equipment involved in transforming energy provided by a power source into useful work done by some machine. In industrial applications, such equipment typically includes an AC motor and a variable speed drive responsible for its control. Aside from direct process control, data gathered in such powertrain applications is also used for higher-level supervisory tasks and condition monitoring. The goal in this use case is to leverage IoT-devices, 5G telecommunication and cloud platforms to utilise novel ideas in the area of data engineering, analytics and condition monitoring.

ABB has two example factories, A and B. Each factory has 3 powertrains, from which sensor data is gathered to a gateway device. The goal is to create a holistic view of the condition and status of each powertrain, especially the drive unit (device that controls the motor) and the motor itself. Instead of using a traditional data-siloed site-specific approach, a decentralised and federated approach is taken, leveraging the IoT-NGIN paradigm and technologies.

Goal 1: A condition monitoring application needs to be able to access the sensor data gathered from powertrains (located at any site) in order to produce analytics results that can be used to monitor the condition of the devices.

- Only the device/site owner (ABB) should have access to this data.
- The Solution should be scalable to support the addition of new devices and sites.
- The Application needs to be able to access the data sources for each powertrain.
    - Need a systematic approach to crawl for and access data endpoints programmatically.

One of the available hardware setups in the powertrain lab is depicted in Figure 9.1.

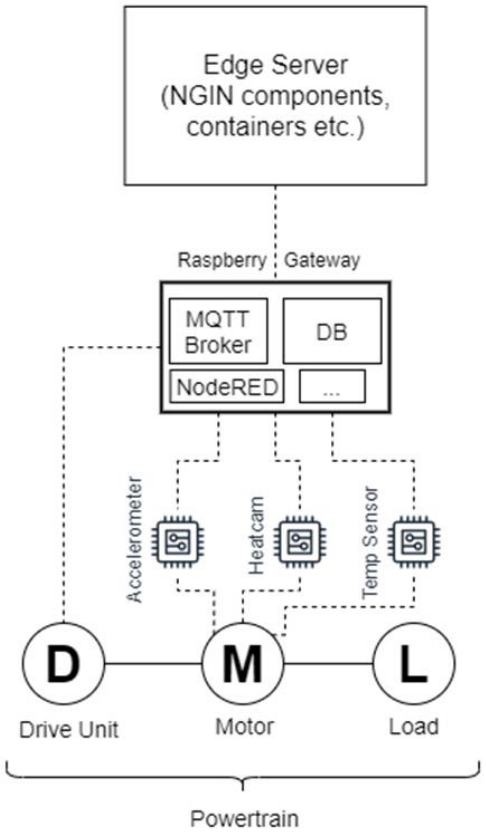

Figure 9.1 - A laboratory setup for a powertrain.

The sensors and devices themselves are not directly capable of running any additional software. Thus, data is gathered to a RaspberryPi or Cassia gateway using available device-specific protocols (OPC DA/UA, plain Transmission Control Protocol (TCP) byte stream, MQTT, File Transfer Protocol (FTP)). The data can then be accessed from the gateway using any protocol of choice. Currently, the gateway device is running Node-RED which can be used to easily create endpoints of preferred protocol/format e.g. MQTT. Descriptions of the devices are given in Table 9.1.

Table 9.1 - Device descriptions in the powertrain use case.

Device	Description
Drive Unit	Acts as a sensor / data source. Has multiple observable signals available related to both the drive's internal operation and motor control. Currently, live data is being sent to the gateway (motor speed, load, voltage, temperature, torque) via 4G. Update interval depends on the signal in question (1s - multiple minutes). Data is sent in bursts / patches and buffered in the drive between transmissions.
Heatcam	A thermal camera that sends data to the gateway as a 2D heatmap / matrix via MQTT (a simple array of values). Currently, uses Node-RED to visualise heatmap to users.
Smart Sensor	A wireless Bluetooth Low Energy (BLE) sensor attached to the side of the motor, which measures vibration and temperature and calculates KPI values. Data is fetched from Cassia gateway using OPC UA.
PLC (accelerometers & temperature sensors)	'Traditional' temperature and accelerometer sensors that are operated using a Programmable Logic Controller (PLC) device. Temperature can be read using Open Platform Communications Data Access (OPC DA). Acceleration data can be fetched from the PLC device as WAV-files using FTP. Accelerometer measurements can be triggered via OPC DA.
RaspberryPi	The gateway device used to collect the data from the various data sources. Currently, uses Node-RED to implement most of the data processing functionality.

The various devices are connected to the gateway using a private 4G network and additional 4G capable gateway modems are used where needed (most sensor devices do not have built-in 4G/5G capabilities).

Twin description documents are created for the powertrains and sensors of the use case. The documents are used in application development, abstracting the underlying protocols used for a specific powertrain setup. The resulting twin description view of the use case is depicted in Figure 9.2 where Powertrain 3 and especially motor M81 have been described in detail.

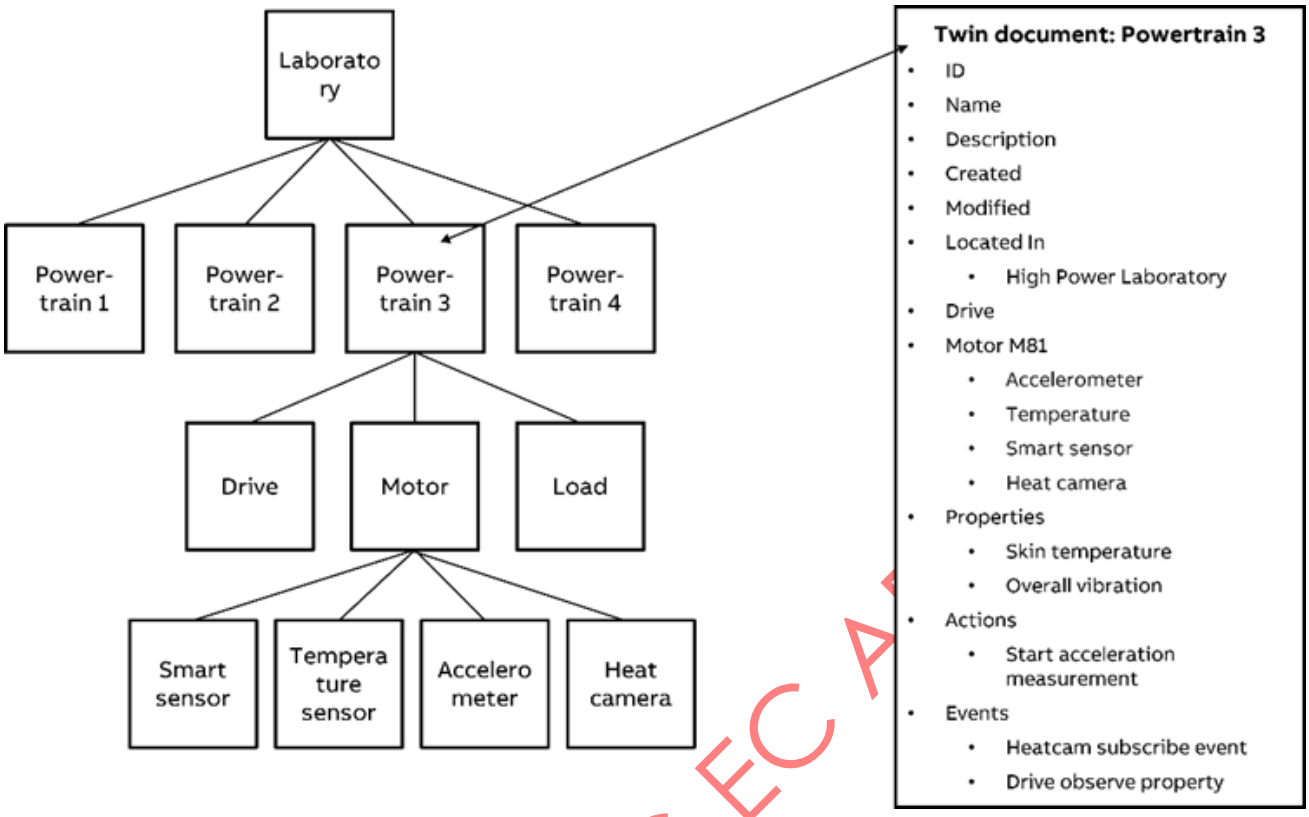

Figure 9.2 - Description of network of Digital Twin documents.

There are several kinds of digital services related to Digital Twin services mentioned in Figure 3.2. To get an idea of possibilities, three different services have been shortly described below:

1. Node-RED
   - Node-RED visualises sensor data from heat camera and smart sensor. The user is able to monitor temperature behaviours of electric motors, for example.
2. Powertrain sensor interface
   - This is an example of condition monitoring of powertrains. The user is able to get real-time data from real installation. With this kind of service, operating parameters can be followed, indicating the status of the powertrain.
3. Digital product
   - This kind of digital service provides access to the digital simulation model of a real powertrain. The simulation model is accessible from a dedicated portal [7] and it can be executed in parallel with a real power train. Normally, the simulation model is executed in a separate environment. The simulation model is able to produce additional data which can be used to indicate maintenance of the powertrain, to optimise the powertrain in operation etc.

Usage of Digital Twin approach is illustrated in Figure 9.3. The following user actions have been recognized:

[7] https://new.abb.com/drives/software-tools/virtual-commissioning-for-drives

1. The user is present at the powertrain site and has physical access to the powertrain.
2. The user is using a mobile device (i.e. cellular phone) to open an application to scan QR code attached to a system or component. The user is able to select a specific powertrain or component if several are available.
3. The user gives necessary data to be identified to get access to the documentation.
4. The user has a view on the selected powertrain asset.
5. The user is able to check the content of the digital document of the selected powertrain. Information is available in the structured way according to Figure A.2. The Metadata of powertrain can be examined, and different kinds of digital services are accessible via links in the documentation.
6. The user selects one of available services, for example, temperature monitoring of an electric motor. Additional identification is maybe required at this point, depending on the service.
7. Instead of monitoring real-time data, the user can switch to virtual monitoring of the electric motor. The user must select a service providing access to the simulation model of the selected powertrain. Again, additional identification is potentially required at this point, depending on the service.

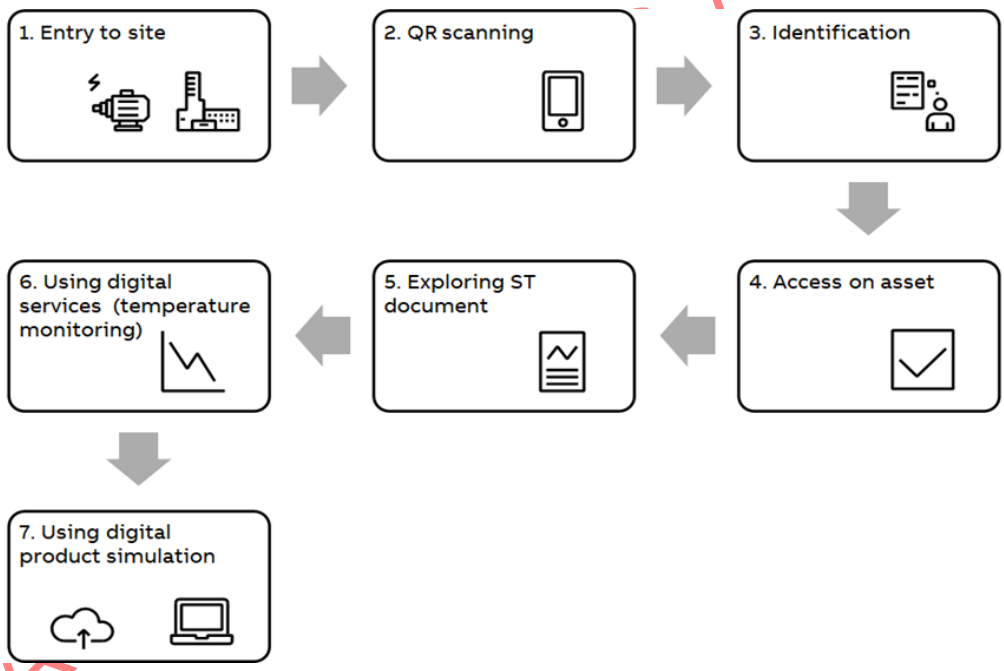

Figure 9.3 - Flow chart of user actions using Digital Twin of a powertrain.

One typical use case is that the user wants to check the status of the electric motor in operation. Temperature of the motor is a good indicator about the status of the motor. Node-RED view on temperature behaviour of selected motor is shown in Figure 9.4. The user is able to check real-time temperature of the motor, but a histogram is also available. The Temperature data can be used to indicate the following things:

1. Data can be used to check if there are any environmental changes visible
2. Need for maintenance actions can be checked, e.g. if predefined temperature limits have been crossed
3. Comparison between different installation can be made
4. Malfunction of sensors can be indicated

Generally, digital services like temperature monitoring can be seen as additional value for different stakeholders of powertrain. Powertrain operators, maintenance service, manufacturers and so on can benefit from this kind of service.

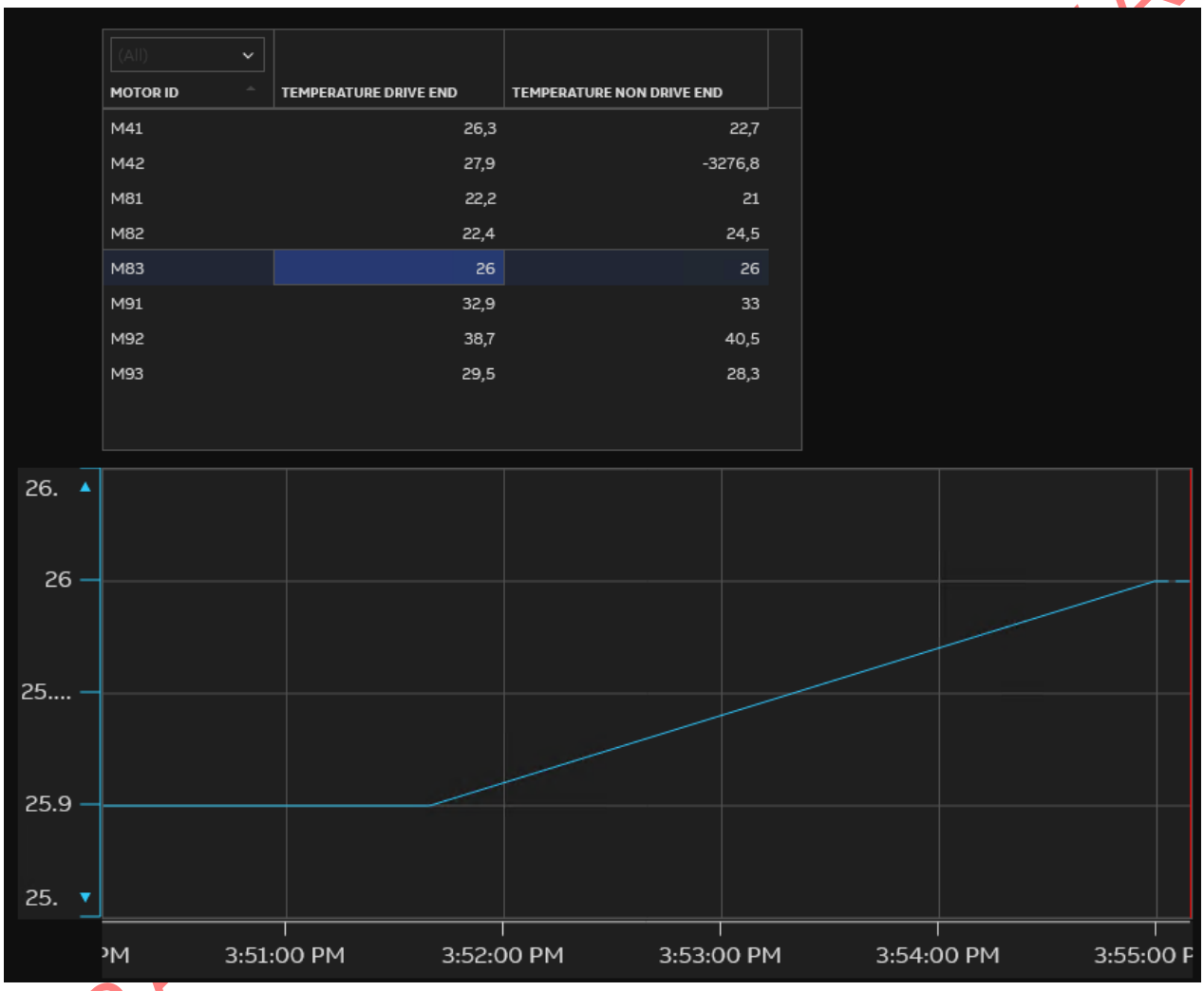

Figure 9.4 - View on temperature of electric motor.