



Enershare

The Energy Data Space for Europe

European Common Energy Data Space Framework Enabling Data Sharing - Driven Across – and Beyond – Energy Services

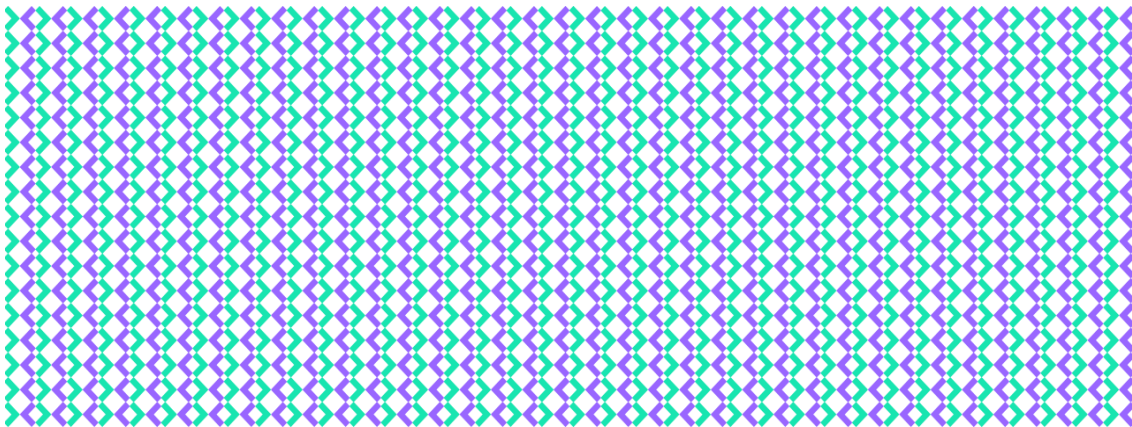
enershare.eu



Enershare has received funding from [European Union's Horizon Europe Research and Innovation programme](#) under the Grant Agreement No 101069831

D5.1 ENERSHARE Data Value Stack

Alpha version



Enershare has received funding from [European Union's Horizon Europe Research and Innovation programme](#) under the Grant Agreement No 101069831

Publication details

Grant Agreement Number 101069831

Acronym ENERSHARE

Full Title	European Common Energy Data Space Framework Enabling Data Sharing-Driven Across — and Beyond — Energy Services
Topic	HORIZON-CL5-2021-D3-01-01 'Establish the grounds for a common European energy data space'
Funding scheme	HORIZON-IA: Innovation Action
Start Date	Jul 1, 2022
Duration	36 months
Project URL	enershare.eu
Project Coordinator	Engineering
Deliverable	D5.1 – ENERSHARE Data Value Stack (Alpha version)
Work Package	WP5 – Data and Services Marketplaces for data sharing value creation and energy compensation
Delivery Month (DoA)	M12
Version	1.0
Actual Delivery Date	October 16, 2023
Nature	Report
Dissemination Level	PU
Lead Beneficiary	ENG



Enershare has received funding from [European Union's Horizon Europe Research and Innovation programme](#) under the Grant Agreement No 101069831

Authors	Diego Arnone (ENG), Lorenzo Cristofori (ENG), Marzia Mamma (ENG), Alessandro Rossi (ENG), Silvia Clementina Santoro (ENG), Apostolos Kapetanios (ED), Konstantinos Kotsalos (ED), José Andrade (INESC-TEC), Ricardo Bessa (INESC-TEC), Fábio Coelho (INESC-TEC), André Garcia (INESC-TEC), Carla Gonçalves (INESC-TEC), Maarten Kollenstart (TNO), Arjan Stoter (TNO), Sonia Bilbao (TECNALIA), Adelaida Lejarazu (TECNALIA), Sonia Jimenez (IDSA)
Quality Reviewer(s)	Vassilis Sakas (ED), Arjan Stoter (TNO)
Keywords	Marketplace, data value stack, data service, trading



Document History

Ver.	Date	Description	Author	Partner
0.1	May 8, 2023	ToC	Diego Arnone	ENG
0.2	May 15, 2023	Final ToC	Diego Arnone	ENG
0.4	September 10, 2023	First Draft	Diego Arnone, Lorenzo Cristofori, Marzia Mammina, Alessandro Rossi, Silvia Clementina Santoro, Apostolos Kapetanios, Konstantinos Kotsalos, José Andrade, Ricardo Bessa, Fábio Coelho, André Garcia, Carla Gonçalves, Maarten Kollenstart, Arjan Stoter, Sonia Bilbao, Adelaida Lejarazu, Sonia Jimenez	ENG, ED, INESC-TEC, TNO, TECNALIA, IDSA
0.5	September 20, 2023	Draft, updated contribution	André Garcia, Ricardo Bessa, Fábio Coelho, Sonia Bilbao	INESC-TEC, TECNALIA
0.6	September 12, 2023	Complete Draft	Marzia Mammina, Silvia Clementina Santoro	ENG
0.7	October 6, 2023	Consolidated	Marzia Mammina, Silvia Clementina Santoro	ENG
0.8	October 12, 2023	Reviewed	Vassilis Sakas, Arjan Stoter	ED, TNO
0.9	October 13, 2023	Release Candidate	Marzia Mammina	ENG
0.99	October 16, 2023	Quality Checked	Massimo Bertoncini	ENG
1.0	October 16,	Final version	Massimo Bertoncini	ENG



2023

Disclaimer

The sole responsibility for the content of this publication lies with the authors. It does not necessarily reflect the opinion of the European Union. Neither the CINEA nor the European Commission is responsible for any use that may be made of the information contained therein.



Enershare has received funding from [European Union's Horizon Europe Research and Innovation programme](#) under the Grant Agreement No 101069831

Table of Contents

1	Introduction	19
1.1	About the project.....	19
1.2	About this document	19
1.3	Intended audience	19
2	Methodology	21
2.1	Actors and Roles	21
2.2	Use Cases	23
2.3	Requirements	24
2.4	Architecture.....	27
3	Use Cases	30
3.1	(Marketplace Participant) Register to the Marketplace.....	30
3.2	(Marketplace Administrator, Marketplace Participant) Login to the Marketplace.....	30
3.3	(Marketplace Administrator, Marketplace Participant) Logout of the Marketplace	31
3.4	(Marketplace Administrator) Validate registration to the Marketplace	32
3.5	(Marketplace Participant) View/Update Profile	32
3.6	(Marketplace Administrator) Validate updates to profiles	33
3.7	(Marketplace Participant) Delete registration to the Marketplace.....	33
3.8	(Marketplace Participant) View transaction history on the Marketplace	34
3.9	(Marketplace Participant as Seller) Publish Dataset.....	35
3.10	(Marketplace Participant as Buyer) Purchase Dataset	35
3.11	(Marketplace Participant as Seller) Publish Data Services	36
3.12	(Marketplace Participant as Buyer) Purchase Data Services	37
3.13	(Marketplace Participant as Seller) Publish Charging Station Availability.....	38
3.14	(Marketplace Participant as Buyer) Purchase Charging Station Availability	38
3.15	(Marketplace Participant as Seller) Publish Apps.....	39
3.16	(Marketplace Participant as Seller/Buyer) Purchase Apps	40
3.17	(Marketplace Participant as Energy Consumer) Modify Energy Contract Conditions....	41
3.18	(Marketplace Participant as Energy Community Promoter) Modify Distribution Coefficients in an Energy Community	42



3.19 (Marketplace Participant as Auction Promoter) Publish Cross-domain Services.....	43
3.20 (Marketplace Participant as Auction Bidder) Propose exchange to obtain a Cross-domain Service.....	44
3.21 (Marketplace Participant as Auction Promoter) Selects best offer as exchange for a Cross-domain Service	44
3.22 (Marketplace Participant as Auction Bidder) Access Cross-domain Service auction result	45
3.23 (Marketplace Participant as Barter Data Producer/Consumer) Exchange Data/Collaborative Data Analytics Services	46
3.24 (Marketplace Participant as Buyer) Rate Asset/Seller.....	47
3.25 (Marketplace Participant as Seller) Visualise Ratings.....	48
4 Requirements	49
5 Static logical view of the Architecture	56
5.1 ENERSHARE Marketplace Graphical User Interface (GUI)	56
5.2 AppStore	57
5.3 Clearing House.....	59
5.4 Metadata Broker.....	63
5.5 Multi-assets Marketplace	64
5.6 Data Monetization and Barter Sharing Incentive Module.....	65
5.6.1 Data Contribution Methods.....	65
5.6.2 Data Types	65
5.6.3 Market Functions.....	66
5.6.4 Collaborative Forecasting	66
5.6.5 Market Engine.....	66
5.6.6 Secure Fund Management through Market Wallet.....	66
5.6.7 Future advancements	66
5.7 Blockchain.....	67
6 Existing implementations	69
6.1 AppStore	69
6.1.1 Actors.....	69
6.1.2 Scenarios and Diagrams.....	70
6.1.3 App Store Architecture	74
6.2 Clearing House.....	76



6.3	Metadata Broker.....	78
6.3.1	TSG Components	78
6.3.2	TSG Metadata Broker	79
6.3.3	Message flows	80
6.3.3.1	Publish Self-Description	80
6.3.3.2	Query self-descriptions.....	81
6.4	Data Monetization and Barter Sharing Incentive Module.....	82
6.4.1	Input and Output Data Format	82
6.4.2	Data Format	82
6.4.3	API – Requests and Structures.....	83
6.4.3.1	Session Bid Request	84
6.4.3.2	Data Management Request.....	87
6.4.4	Running the Service	88
6.4.4.1	Desktop Application.....	88
6.4.4.2	Developers	88
6.4.4.3	System administrator (Market Operator).....	88
6.4.5	Technology Readiness Level (TRL)	89
6.4.6	Software Details.....	90
6.4.7	Integration with ENERSHARE Data Space	91
6.5	Blockchain.....	95
6.5.1	Hardware Infrastructure	95
6.5.2	Ethereum and public testnets.....	95
6.5.3	Hyperledger and private chains.....	97
6.5.4	Smart Contracts	98
6.5.5	ENERSHARE Token	99
6.5.6	Proof of Existence contract.....	101
7	Conclusions	105
Appendix: Data Monetization and Barter Sharing Incentive Module		106
Data Contribution		106
Data types		106
Market Functions		107





Market Agents Registration	107
Market Agents Authentication	107
Agents Portfolio Management	108
Agents Market Participation	108
Measurements upload.....	109
Forecasts download.....	109
Market Engine	109
Market Session Management.....	110
Agents Bids Validation	110
Agent Data Validation.....	111
Collaborative Forecasting	111
Revenue definition and distribution.....	111
Market Wallet	112
Bid Validation.....	112
Secure Fund Management.....	112
Transparent Bid Validation	112
Payment Function.....	113
Integration with Blockchain or Distributed Ledger Technology	113
Market Agent (Desktop application scope)	113
Local Data Parsers.....	113
Local Cryptocurrency Wallet.....	114



List of Figures

Figure 1: General process for requirement management.....	27
Figure 2: The 4+1 Architectural View Model	28
Figure 3: ENERSHARE Marketplace Architecture	56
Figure 4: App Store lifecycle workflow	58
Figure 5: Logic and functions of the IDS Clearing House	61
Figure 6: Logical view of the Metadata Broker.....	63
Figure 7: App Store boot process sequence diagram	73
Figure 8: App store publish data process sequence diagram	74
Figure 9: App Store Architecture	75
Figure 10: Clearing House Architecture.....	76
Figure 11: OneNet data exchanges Timeline providing primary meta-data logging.....	78
Figure 12: Publish self-description message flow.....	81
Figure 13: Query self-description message flow	82
Figure 14: API documentation overview	83
Figure 15: Traditional server-client communication vs decentralized communication	92
Figure 16: Registration and validation of data by means of the blockchain	103
Figure 17: Overview architecture and available interactions for the transfer of data.....	106
Figure 18: Bid validation process.....	112



List of Tables

Table 1: Actor, Roles, and Responsibilities	22
Table 2: Use Case template	24
Table 3: Functional requirements definition template	26
Table 4: Register to the Marketplace Use Case	30
Table 5: Login to the Marketplace Use Case	30
Table 6: Logout of the Marketplace Use Case	31
Table 7: Validate registration to the Marketplace Use Case	32
Table 8: View/Update Profile Use Case	32
Table 9: Validate updates to profiles Use Case	33
Table 10: Delete registration to the Marketplace Use Case	34
Table 11: View transactions history on the Marketplace Use Case	34
Table 12: Publish Dataset Use Case	35
Table 13: Purchase Dataset Use Case	35
Table 14: Publish Data Services Use Case	36
Table 15: Purchase Data Services Use Case	37
Table 16: Publish Charging Station Availability Use Case	38
Table 17: Purchase Charging Station Availability Use Case	38
Table 18: Publish Apps Use Case	39
Table 19: Purchase Apps Use Case	40
Table 24: Modify Energy Contract Conditions Use Case	41
Table 25: Modify Distribution Coefficients in an Energy Community Use Case	42
Table 26: Publish Cross-domain Services Use Case	43
Table 27: Propose exchange to obtain a Cross-domain Service Use Case	44
Table 28: Selects best offer as exchange for a Cross-domain Service Use Case	44
Table 29: Access Cross-domain Service auction result Use Case	45
Table 30: Exchange Data/Collaborative Data Analytics Services Use Case	46
Table 31: Rate Asset/Seller Use Case	47
Table 32: Visualise Ratings Use Case	48
Table 33: Functional requirements definition	49
Table 34: App Store System Actors	69



Table 35: App Store Boot Process scenario	70
Table 36: App Store Publish Data Apps scenario.....	71
Table 37: App Store Browse and Retrieve Data Apps scenario	72
Table 38: API integration bid script example.....	84
Table 39: API requests examples for uploading and downloading of data	87
Table 40: Comparison between the main features of Ethereum and Hyperledger	97
Table 41: JSON array of registered data example	104



List of Acronyms

ABAC	Attribute-Based Access Control
API	Application Programming Interface
APM	Application Performance Monitoring
App	Application
B2B	Business to Business
B2C	Business to Consumer
CH	Clearing House
CI/CD	Continuous Integration and Deployment
CRUD	Create, Read, Update and Delete
CSS	Cascading Style Sheets
CSV	Comma-Separated Values
DAO	Decentralized Autonomous Organization
Dapps	Decentralized Applications
DAPS	Dynamic Attribute Provisioning Service
DC	Data Consumer
DeFi	Decentralized Finance
DLT	Distributed Ledger Technology
DP	Data Provider
DTM	Dynamic Trust Management
DSO	Distribution System Operators
EC	European Commission



ERC20	Ethereum Request for Comments 20
E \odot T	ENERSHARE Token
EVM	Ethereum Virtual Machine
FAQ	Frequently Asked Questions
GPL	General Public License
GUI	Graphical User Interface
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol over Secure Socket Layer
IBFT	Istanbul Byzantine Fault Tolerant
IDS	International Data Spaces
IDSA	International Data Spaces Association
IEEE	Institute of Electrical and Electronic Engineers
JSON	JavaScript Object Notation
MIT	Massachusetts Institute of Technology
MTU	Market Time Unit
NFT	Non-Fungible Token
PBAC	Policy-Based Access Control
PIN	Personal Identification Number
PoA	Proof of Authority
PoE	Proof of Existence
PoLP	Principle of Least Privilege
PoS	Proof of Stake



PoW	Proof of Work
SHA-3	Secure Hash Algorithm 3
SSL	Secure Sockets Layer
TLS	Transport Layer Security
TRL	Technology Readiness Level
TSG	TNO Security Gateway
UC	Use Case
UML	Unified Modeling Language
WP	Work Package
XML	Extensible Markup Language



Executive summary

The ongoing energy system digitization is making available an enormous amount of data, paving the way for data sharing-enabled services, which may contribute to system-level increased efficiency and hence facilitate the energy transition.

The ENERSHARE project aims to extend, demonstrate, and validate, through field pilots, conceptual Data Space architectures and technology components and intends to demonstrate how different roles along Data Value Chain can emerge in the energy sector, with a view to develop and deploy an overall reference implementation for a common European Energy Data Space.

The ENERSHARE Data Space will include a set of shared capabilities and services aimed to facilitate and enable the access, sharing, and trading of data assets (datasets, data services) among a variety of data infrastructures.

This Deliverable 5.1 ENERSHARE Data Value Stack (Alpha version) is the first result of the joint activities of WP5, namely Task 5.1 “Publication and data marketplace services”, Task 5.2 “Data usage accounting (Clearing House)”, and Task 5.3 “Tokenised Appstore marketplace and smart contracts for heterogeneous data vs energy services compensation”.

WP5 evolves and adapts the actual implementations of related IDSA Building Blocks to deliver an energy-adapted stack of common data value services within the ENERSHARE Data Space. It will provide energy-specific developments for: (1) Data Broker for metadata publishing, search and discovery; (2) Clearing House for transactions accounting and settlement; (3) App-store-like Marketplace for data providers assets and/or data service providers; (4) Providing a cross-stakeholder P2P digital marketplace as further incentive for energy end users to share their data and being eventually rewarded with suitable heterogeneous assets (e.g. device maintenance services, prioritised network outage fixing).

WP5 will provide the ENERSHARE Marketplace structured as below reported:

- A shop window, a shelf on top of which asset providers may publish the assets they want to sell (such as dataset, data service, Apps, charging station availability) and interested consumers may buy them.
- A session-based electricity marketplace. Market sessions can be day-ahead/intraday and refer to a specific electricity delivery time period. While a market session is running, the electricity buyers/sellers can place bids/offers. When the market session ends, the market is cleared: all valid supply offers are put in increasing price order on an aggregate supply curve and all valid demand bids are put in decreasing price order on an aggregate demand curve. The intersection of the two curves determines the market



equilibrium, which gives the clearing price, at which all accepted bids and offers are remunerated, and the overall quantity of energy traded in the session. The market session results are made available to participants.

- A barter of data: valuable data for a specific service (e.g., load and renewable energy time series forecasting) are distributed across multiple owners/devices and monetary and non-monetary (barter) incentive mechanisms foster data sharing and enable collaborative data analytics.
- The cross-domain services/assets auction. Who wants to provide any assets open an auction characterized by a specific duration and during which other participants can propose any asset in exchange. As soon as the auction is closed, the user who started the auction can decide to accept (by choosing what they prefer) or to reject all the counterproposals.

Moreover, the ENERSHARE Marketplace allows energy consumers to ask energy trader for new contract conditions. Finally, through the Marketplace, members of an Energy Community may propose a modification of the distribution coefficients in an energy community (flexibility).

The purpose of this document is to present the methodological approach and the preliminary steps to design the software architecture (namely the static behaviour), starting from the Use Cases related to ENERSHARE Marketplace framework and the related requirements. The static logical view of the ENERSHARE Marketplace software architecture is presented and main features of its modules are described.

This document also provides a description of ENERSHARE Data Value Stack Alpha Version software release, which contains already existing implementation of those components that will be part of ENERSHARE architecture.



1 Introduction

1.1 About the project

The overall purpose of ENERSHARE is to develop and establish a European Common Energy Data Space that will deploy an **intra-energy** and **cross-sector** interoperable and trusted Energy Data Ecosystem. Private consumers, business (energy and non-energy) stakeholders and regulated operators will be able to **access, share, and reuse**, based upon voluntary agreements or legal obligations, large sources of currently fragmented and disseminated data and data-driven cross-value chain (energy and non-energy) services.

1.2 About this document

This Deliverable 5.1 ENERSHARE Data Value Stack (Alpha version) is the first result of the joint activities of WP5, namely Task 5.1 “Publication and data Marketplace services”, Task 5.2 “Data usage accounting (Clearing House)”, and Task 5.3 “Tokenised Appstore Marketplace and smart contracts for heterogeneous data vs energy services compensation”.

The purpose of this document is to present the methodological approach and the preliminary steps to design the software architecture), starting from the Use Cases related to ENERSHARE WP5 framework and the related requirements.

The remainder of this deliverable is structured as follows:

- In Chapter 2, the methodological approach is explained.
- Chapter 3 presents the Use Cases.
- In Chapter 4, the high-level functional requirements are listed.
- Chapter 5 presents the ENERSHARE software architecture and describe the main components.
- In Chapter 6, the existing implementations that will be released as ENERSHARE Data Value Stack Alpha Version are described.
- In Chapter 7, conclusions are given.

1.3 Intended audience

This document is marked as "Public"; thus, beyond being consulted by the consortium partners and the European Commission (EC) representatives tasked with reviewing the project, it will be published on the project website and made available for a wider access. Making project



deliverables and reports publicly available can also help to foster collaboration, promote transparency, and facilitate the adoption of new technologies and practices.



2 Methodology

This section explains the methodology that is used to formalize the use cases, defines requirements, and create the architecture design.

2.1 Actors and Roles

A key concept of our method is the definition of actors and roles.

An **actor** is an entity that is a participant in a process and/or a user of a system. An actor should be given a name that reflects its interaction with the business, i.e., the business service that it interacts with. The name should be a noun and be applicable to any person—or any information system—playing the part of the actor.

A **role** is a position that an actor has in a process. According to their role and responsibilities, actors are expected to perform certain tasks. A role should be given a name that reflects its specific business activity. The name should be a noun for functional roles and reflect to responsibilities of the activity.

A responsibility is the set of tasks, activities or actions an actor is expected to perform, or allowed to do, according to the role. Example: The responsibility of an Elderly Woman (Actor) as Shopper (Role) on a Webshop (System) is to Register for the Webshop (Task), Add Items to A Cart (Task) and To Pay For the Items (Task).¹

An actor, associated with a "user" (person) interacts with the "system" through one or more roles. A role is equated to -and defined by- the behaviour (and by extension the responsibilities) of the associated activity. A role can be assumed by any kind of actor, including humans and systems. A role may be associated with only one activity, but there may be multiple roles associated with the activity. An actor can be associated with multiple persons or systems.

The checklist below defines things to look for when defining actors, to ensure that they are well-defined, that the model is complete and that it is consistent².

- *Have all the actors and roles been identified?*
- *Is each role associated with at least one business activity?*
- *Does each business service (process) have an associated actor?*
- *Are there at least two people who would perform the role (activity) of a particular actor?*

¹ <https://www.dragon1.com/resources/actors-roles-responsibilities>

² https://www.unified-am.com/UAM/UAM/guidances/checklists/uam_actor_B0306C90.html





- *Do any actors play similar roles?*
- *Do two actors play the same role with a business activity?*
- *Does a particular role/actor use the business activity in several (completely different) ways?*
- *Do the actors have intuitive and descriptive names?*

The Table 1 below shows actors roles and main responsibilities in the ENERSHARE Marketplace.

Table 1: Actor, Roles, and Responsibilities

Actor	Role	Responsibilities
Marketplace Administrator	Operator	<ul style="list-style-type: none"> • validate participants registration • validate profile updates
Marketplace Participant	Seller	<ul style="list-style-type: none"> • register • put up for sale: <ul style="list-style-type: none"> ○ Dataset ○ Data Service ○ Charging Station Availability ○ Apps • view transaction history
Marketplace Participant	Buyer	<ul style="list-style-type: none"> • buy: <ul style="list-style-type: none"> ○ Dataset ○ Data Service ○ Charging Station Availability ○ Apps • view transaction history
Marketplace Participant	Energy Consumer	ask for a new electricity contract
Marketplace Participant	Energy Trader	offer a new contract
Marketplace Participant	Energy Community Promoter	propose a new distribution of the coefficients in the Energy Community
Marketplace Participant	Energy Community Approver	accept or refuse the request
Marketplace Participant	Electricity Market Participant	post bid/offer in a session of the electricity market (day ahead or intraday)
Marketplace Participant	Electricity Market Operator	schedule session
Marketplace Participant	Auction Promoter	start an auction, where an asset that is already published in the Marketplace is offered (dataset, data service, etc.)
Marketplace Participant	Auction Bidder	offer an asset as a counteroffer in an auction
Marketplace Participant	Barter Data Consumer	submit requests (necessary data and maximum price) for a dataset
Marketplace Participant	Barter Data Producer	<ul style="list-style-type: none"> • agree with terms and conditions for data provision and submit offer



		<ul style="list-style-type: none"> • receive the funds in case of payment
Marketplace Participant	Barter Operator	clear the market and define data exchange, and set a fair payment distribution in case of data monetization

2.2 Use Cases

A Use Case (UC) is a definition of a specific business objective that a system needs to accomplish. This is done by describing the various actors that exist outside of the system, together with the specific interactions the actors have with the system to accomplish the business objective³.

Essentially, UCs answer three questions:

- Who is going to use a specific product?
- What will it be used for?
- How are they going to use it?

The key to produce an effective UC is to carefully consider the flow of events for a typical user given a specific business objective.

UCs can be as high-level or detailed as they need to be according to the audience. Business UCs, also known as Abstract-Level UCs, are written in a technology-agnostic manner, simply referring to the high-level business processes being described and the various external actors that take part in the process. A Business UC defines the sequence of actions that the business needs to perform to provide a meaningful and observable result to the external actors. System UCs, also known as implementation UCs, have a higher level of detail than the business use cases and refer to specific processes that will be carried out by different parts of a system. It would then describe the interactions of the various actors with the system in carrying out the end-to-end process.

Typically, the high-level Business UCs are drafted first, and, successively, they will be broken down into one or more lower-level System UCs.

One artifact related to a UC is the business scenario. Similarly, to a UC, a business scenario describes what the external actor seeks to accomplish for a specified business process; however, unlike a UC, which is a step-by-step enumeration of the tasks carried out during a process, a scenario is much more free-form.

³ <https://www.inflectra.com/Ideas/Topic/Use-Cases.aspx>



Table 2 shows the template for Business UCs used within ENERSHARE WP5 and describe the elements of the UC required to understand its definition, including actors, conditions, trigger events, etc.

Table 2: Use Case template

Use Case <#. #>: <Use Case Name (best use a verb of action)>	
Brief Description	<A brief description of the use case, no longer than a small paragraph; preferably keep it one clear goal per use case>
Actor(s)	<All users interacting with the ENERSHARE System; preferably identified by role name>
Priority	<Depending on how many use cases we end up with we may need to prioritize them based on importance to meet the project objective: high (must do) medium (should do, time permitted) low (nice to do, but most likely not)>
Trigger	<Concrete actions by users interacting with the ENERSHARE System to initiate use case>
Pre-conditions	<System State prior to launching the use case >
Post-conditions	<System State expected after the use case has been completed >
Basic Flow	<Describe the basic flow of events during use case Step 1. Step 2. ... Step N. >
Alternate Flow(s)	<If applicable, provide the steps of less common user/system interactions>
Exception Flow(s)	<Anything that may happen that would prevent the user from achieving their goal>

2.3 Requirements

Concerning requirements, we distinguish between *functional* and *non-functional* requirements as⁴:

- i. *Functional requirements* describe something the system must do. If the system does not meet a functional requirement it will fail, since it will not be able to achieve something, it must do to operate properly. Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionalities that define what a system is supposed to accomplish. Functional requirements are supported by non-functional requirements.

⁴ <https://enkonix.com/blog/functional-requirements-vs-non-functional/>





- ii. *Non-functional requirements* elaborate a performance characteristic of the system. They are focused on how the system goes about delivering a specific function; non-functional requirements do not have an impact on the functionality of the system, but they do impact on how it will perform. They are constraints imposed on the design or implementation (such as performance, security, cost, robustness, portability, reliability, interoperability) of the overall system. Meeting at least some non-functional requirements is important in a well performing system.

According to IEEE 29148-2011:

"A well-formed requirement is a statement that:

- I. can be verified,*
- II. has to be met or possessed by a system to solve a stakeholder problem or to achieve a stakeholder objective,*
- III. is qualified by measurable conditions and bounded by constraints, and*
- IV. defines the performance of the system when used by a specific stakeholder or the corresponding capability of the system, but not a capability of the user, operator, or other stakeholder.*

It is important to agree in advance on the specific keywords and terms that signal the presence of a requirement. A common approach is to stipulate the following:

- I. Requirements are mandatory binding provisions and use 'shall'.*
- II. Statements of fact, futurity, or a declaration of purpose are non-mandatory, non-binding provisions and use 'will'. 'Will' can also be used to establish context or limitations of use. However, 'will' can be construed as legally binding, so it is best to avoid using it for requirements.*
- III. Preferences or goals are desired, non-mandatory, non-binding provisions and use 'should'.*
- IV. Suggestions or allowances are non-mandatory, non-binding provisions and use 'may'.*
- V. Non-requirements, such as descriptive text, use verbs such as 'are', 'is', and 'was'. It is best to avoid using the term 'must', due to potential misinterpretation as a requirement.*
- VI. Use positive statements and avoid negative requirements such as 'shall not'.*
- VII. Use active voice: avoid using passive voice, such as 'shall be able to select'.*
- VIII. [Condition] [Subject] [Action] [Object] [Constraint]*

Examples of requirements syntax:

[Condition] [Subject] [Action] [Object] [Constraint]

EXAMPLE: *When signal x is received [Condition], the system [Subject] shall set [Action] the signal received bit [Object] within 2 seconds [Constraint].*

Or

[Condition] [Action or Constraint] [Value]



EXAMPLE: At sea state 1 **[Condition]**, the Radar System shall detect targets at ranges out to **[Action or Constraint]** 100 nautical miles **[Value]**.

Or

[Subject] [Action] [Value]

EXAMPLE: The Invoice System **[Subject]**, shall display pending customer invoices **[Action]** in ascending order **[Value]** in which invoices are to be paid.”

Each requirement is numbered (so as to be uniquely identifiable) and prioritized. The following definitions are intended as a guideline to prioritize requirements:

- High – The requirement is a “must have” as outlined by policy/law
- Medium – The requirement is needed for improved processing, and the fulfilment of the requirement will create immediate benefits
- Low – The requirement is a “nice to have” which may include new functionality

In this document, the high-level functional requirements will be listed. To describe the functional requirements, the template shown in Table 3 will be used.

Table 3: Functional requirements definition template

ReqID	Name	Priority	Derived from	Description
< unique id of the functional requirement >	<requirement name>	high medium low	< use cases>	< well-formed description of functional requirement >

The process for managing requirements is highly dependent on the methodology being used in the project (waterfall, agile, hybrid, etc.). The process that will be applied in ENERSHARE WP5 is represented in Figure 1.



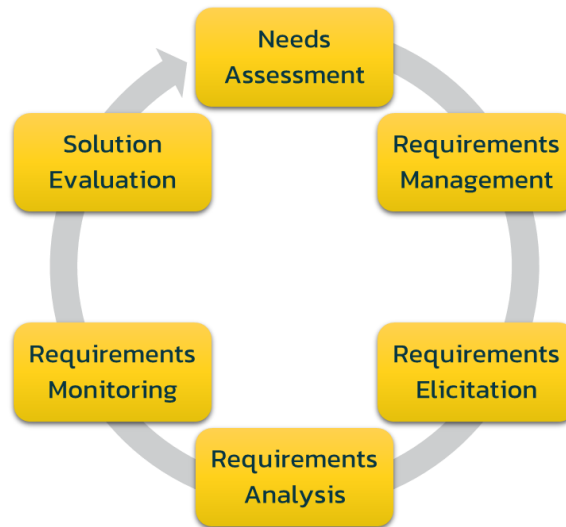


Figure 1: General process for requirement management⁵

Below a description of each phase is given:

- **“Needs**– This is the process of identifying the needs and understanding the underlying business drivers, scope, constraints and context.
- **Requirements Management** – The process of putting in place a process to manage the requirements before they are gathered
- **Requirements Elicitation** – Also called requirements gathering, the process of identifying and researching all of the requirements and needs.
- **Requirements Analysis** – Sometimes called functional analysis, the process of analysing the requirements so far, identifying gaps, duplications, contradictions and areas of further investigation.
- **Requirements Monitoring** – The process of monitoring the requirements as the project is underway to see if they need to **Assessment** explicitly change or whether there were simplifications or misunderstandings that result in clarifications and changes.
- **Solution Evaluation** – Benchmarking the final solution against the requirements to see if there are gaps (for future development), or whether there are missing features or test cases.”⁵

2.4 Architecture

A view, in software architecture, is a representation of one or more structural aspects of a software system architecture and illustrates how the architecture addresses one or more concerns held by one or more of its stakeholders⁶.

⁵ <https://www.inflectra.com/Ideas/Topic/Requirements-Gathering.aspx>

⁶ IEEE std 1471-2000 Recommended Practice for Architectural Description of Software-Intensive Systems



In 1995, Philippe Kruchten, Lead software architect at Hughes Aircraft of Canada, in his paper released that year, *Architectural Blueprints — The “4+1” View Model of Software Architecture*, presented a technique for organizing the description of a software architecture using a set of concurrent “views,” each addressing unique concerns for distinct stakeholders. End-users, developers, system engineers, and project managers all have unique views on the system and, thus, the viewpoints are used to describe the system from their perspectives. This explains why this technique is called the 4+1 Architectural Model Originated.

There are 5 views, but Kruchten decided to call it “4+1”, because, when all other views are finished, the fifth view, the use case view, is effectively redundant. However, the use case view is the basis for all other views: it details the high levels requirements of the system, while the other views detail how those requirements are realised.

The 4+1 views proposed by Kruchten (Figure 2) are below described:

- The **logical view** represents the object model of the design (when an object-oriented design method is used). It shows the functional requirements related to the final user. It is relevant to developers.
- The **process view** describes the concurrent processes within the system. It includes some non-functional requirements such as availability and performance.
- The **development view** focusses on software modules and subsystems.
- The **deployment** (or physical) **view** depicts the mapping of the software onto the hardware and it includes some non-functional requirements such as availability and scalability.
- The **scenarios** (or use case) **view** illustrates the system functionality from the perspective of the external world. All other views use the scenario view to guide them.

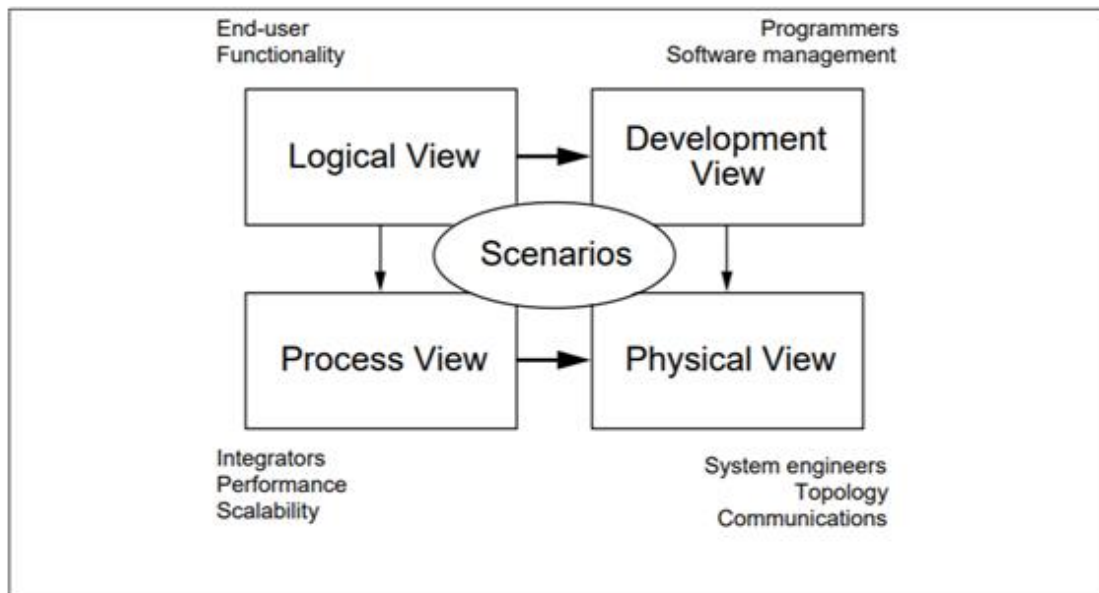


Figure 2: The 4+1 Architectural View Model

In his paper “*Architectural Blueprints - The “4+1” View Model of Software Architecture*”, Kruchten does not make specific references to UML. However, many authors have illustrated



with their “4+1” - Unified Modeling Language (UML) mappings that a different set of UML diagrams exists for each view, which can be useful to convey specific information associated with that view⁷⁸. In addition, some UML diagrams can be used in different ways, by emphasizing different elements present in the diagram, which makes them useful for multiple views. In the following, we will represent the 4+1 views as below described:

- **Logical view** through block diagram.
- **Process view** through sequence diagrams, activity diagrams.
- **Development view** through component diagram.
- **Deployment view** through deployment diagram.
- **Scenarios view** through use case diagrams.

⁷ M. Grgec and R. Muzar, *Role of UML Sequence Diagram constructs in object lifecycle concept*.

⁸ V. Muchandi, *Applying 4+1 View Architecture with UML 2*, 2007.



3 Use Cases

This section illustrates the UCs using the template presented in section 2.2.

3.1 (Marketplace Participant) Register to the Marketplace

Table 4 below describes the (Marketplace Participant) Register to the Marketplace UC.

The Marketplace Participant can register as Seller, Buyer or both.

Table 4: Register to the Marketplace Use Case

Use Case UC_MANAGEMENT_1: Register to the Marketplace	
Brief Description	In order to participate in the Marketplace, the transacting parties must be registered users. Marketplace Participant can register as Seller, Buyer or both and the Marketplace Administrator will later validate the registration.
Actor(s)	Marketplace Participant
Priority	High
Trigger	On demand.
Pre-conditions	The system is installed and active.
Post-conditions	The Marketplace Participant has required an account for actively participating in the Marketplace.
Basic Flow	Step 1. The Marketplace Participant connects to the Marketplace application and selects the registration page. Step 2. The Marketplace Participant inserts personal data. Step 3. The Marketplace Participant submits registration request.
Alternate Flow(s)	
Exception Flow(s)	

3.2 (Marketplace Administrator, Marketplace Participant) Login to the Marketplace

Table 5 below describes the (Marketplace Administrator, Marketplace Participant) Login to the Marketplace UC.

Table 5: Login to the Marketplace Use Case

Use Case UC_MANAGEMENT_2: Login to the Marketplace



Enershare has received funding from [European Union's Horizon Europe Research and Innovation programme](#) under the Grant Agreement No 101069831

Brief Description	The user (Marketplace Administrator, Marketplace Participant) will be prompted to login with their Marketplace account information before they can use the system.
Actor(s)	Marketplace Administrator, Marketplace Participant
Priority	
Trigger	User requests to login
Pre-conditions	The user has a login account.
Post-conditions	The user is logged in to the system. The user has access to the functionalities of the system.
Basic Flow	Step 1. User clicks on “Login”. Step 2. The system prompts the users for their Marketplace account credentials. Step 3. The user enters their Marketplace username and password. Step 4. The system authenticates the user. Step 5. The user gains access to the system functionalities.
Alternate Flow(s)	
Exception Flow(s)	Incorrect credentials

3.3 (Marketplace Administrator, Marketplace Participant) Logout of the Marketplace

Table 6 below describes the (Marketplace Administrator, Marketplace Participant) Logout of the Marketplace UC.

Table 6: Logout of the Marketplace Use Case

Use Case UC_MANAGEMENT_3: Logout of the Marketplace	
Brief Description	The user (Marketplace Administrator, Marketplace Participant) clicks on “Logout” and their session is terminated.
Actor(s)	Marketplace Administrator, Marketplace Participant
Priority	
Trigger	User is done using the web application.
Pre-conditions	The user is logged in. The user no longer wants to be logged in.
Post-conditions	The user is logged out.
Basic Flow	Step 1. The user clicks on the “Logout”. Step 2. The system logs the user out and invalidates the cookie/session. Step 3. The system redirects to the default logout page.
Alternate Flow(s)	
Exception Flow(s)	



3.4 (Marketplace Administrator) Validate registration to the Marketplace

Table 7 below describes the (Marketplace Administrator) Validate registration to the Marketplace UC.

Table 7: Validate registration to the Marketplace Use Case

Use Case UC_MANAGEMENT_4: Validate registration to the Marketplace	
Brief Description	The Marketplace Administrator validates the registration of the Marketplace Participant to the Marketplace, after verifying if eligibility requirements are met.
Actor(s)	Marketplace Administrator as Operator
Priority	High
Trigger	The Marketplace Participant requests registration to the Marketplace.
Pre-conditions	The system is installed and active. The Marketplace Administrator has a login account.
Post-conditions	The Marketplace Participant has a login account and can actively participate in the Marketplace.
Basic Flow	Step 1. The Marketplace Administrator accesses the area where registration requests can be approved. Step 2. The Marketplace Administrator selects a request to approve. Step 3. The Marketplace Administrator checks the eligibility requirements. Step 4. The Marketplace Administrator validates the request.
Alternate Flow(s)	Step 3.1. Marketplace Participant's eligibility requirements are not met; the Marketplace Administrator rejects the registration request.
Exception Flow(s)	

3.5 (Marketplace Participant) View/Update Profile

Table 8 below describes the (Marketplace Participant) View/Update Profile UC.

Table 8: View/Update Profile Use Case

Use Case UC_MANAGEMENT_5: View/Update Profile	
Brief Description	The Marketplace Participants may like to view and/or update their personal profile.
Actor(s)	Marketplace Participant
Priority	Medium
Trigger	The Marketplace Participant logs into the Marketplace application aiming to view and/or update her/his profile.



Pre-conditions	The system is installed and active. The Marketplace Participant has a login account.
Post-conditions	The Marketplace Participant views and/or updates her/his profile.
Basic Flow	Step 1. The Marketplace Participant accesses the profile area in the Marketplace. Step 2. The Marketplace Participant view/update her/his profile.
Alternate Flow(s)	
Exception Flow(s)	

3.6 (Marketplace Administrator) Validate updates to profiles

Table 9 below describes the (Marketplace Administrator) Validate updates to profiles UC.

Table 9: Validate updates to profiles Use Case

Use Case UC_MANAGEMENT_6: Validate updates to profiles	
Brief Description	The Marketplace Administrator validates updates on Marketplace Participant profile, after verifying if eligibility requirements are still met.
Actor(s)	Marketplace Administrator as Operator
Priority	Medium
Trigger	The Marketplace Participant requires to modify her/his own profile.
Pre-conditions	The system is installed and active. The Marketplace Administrator has a login account.
Post-conditions	The Marketplace Participant has a new profile.
Basic Flow	Step 1. The Marketplace Administrator accesses the area where profile update requests can be approved. Step 2. The Marketplace Administrator selects a request to approve. Step 3. The Marketplace Administrator checks if the eligibility requirements are still met. Step 4. The Marketplace Administrator approves the request.
Alternate Flow(s)	Step 3.1. Marketplace Participant's eligibility requirements are not met anymore; the Marketplace Administrator rejects the update request.
Exception Flow(s)	

3.7 (Marketplace Participant) Delete registration to the Marketplace

Table 10 below describes the (Marketplace Participant) Delete registration to the Marketplace UC.





Table 10: Delete registration to the Marketplace Use Case

Use Case UC_MANAGEMENT_7: Delete registration to the Marketplace	
Brief Description	The Marketplace Participants may like to delete their personal profile.
Actor(s)	Marketplace Participant
Priority	Medium
Trigger	The Marketplace Participant logs into the Marketplace application aiming to delete his profile.
Pre-conditions	The system is installed and active. The Marketplace Participant has a login account.
Post-conditions	The Marketplace Participant deleted her/his profile.
Basic Flow	Step 1. The Marketplace Participant accesses the profile area in the Marketplace. Step 2. The Marketplace Participant selects the registration delete button. Step 3. The Marketplace Participant confirm delete operation.
Alternate Flow(s)	
Exception Flow(s)	

3.8 (Marketplace Participant) View transaction history on the Marketplace

Table 11 below describes the (Marketplace Participant) View transaction history on the Marketplace UC.

Table 11: View transactions history on the Marketplace Use Case

Use Case UC_TRANSACTION_1: View transactions history on the Marketplace	
Brief Description	The Marketplace Participants may like to view their transaction history.
Actor(s)	Marketplace Participant
Priority	Medium
Trigger	The Marketplace Participant logs into the market application aiming to view her/his transaction history.
Pre-conditions	The system is installed and active. The Marketplace Participant has a login account.
Post-conditions	The Marketplace Participant knows her/his transaction history.
Basic Flow	Step 1. The Marketplace Participant accesses the transaction history area in the Marketplace. Step 2. The Marketplace Participant selects the transaction for visualising details.
Alternate Flow(s)	



Exception Flow(s)	
-------------------	--

3.9 (Marketplace Participant as Seller) Publish Dataset

Table 12 below describes the (Marketplace Participant as Seller) Publish Dataset UC.

Table 12: Publish Dataset Use Case

Use Case UC_DATASET_1: Publish Dataset	
Brief Description	The Marketplace Participant as Seller inserts descriptive and technical information, including licence, price, and pricing scheme, about the offered dataset in order to make it available on the Marketplace catalogue for selling.
Actor(s)	Marketplace Participant as Seller
Priority	High
Trigger	On demand.
Pre-conditions	The system is installed and active. The Marketplace Participant as Seller has a login account.
Post-conditions	Dataset is available on the Marketplace catalogue.
Basic Flow	Step 1. The Marketplace Participant as Seller accesses the area for publishing a dataset. Step 2. The Marketplace Participant as Seller provides the descriptive and technical information for the dataset.
Alternate Flow(s)	
Exception Flow(s)	

3.10 (Marketplace Participant as Buyer) Purchase Dataset

Table 13 below describes the (Marketplace Participant as Buyer) Purchase Dataset UC.

Table 13: Purchase Dataset Use Case

Use Case UC_DATASET_2: Purchase Dataset	
Brief Description	The Marketplace Participant as Buyer wants to purchase datasets.
Actor(s)	Marketplace Participant as Buyer, Market Participant as Seller
Priority	High
Trigger	The Marketplace Participant as Buyer interacts with ENERSHARE Marketplace to discover interesting datasets available for his/her needs.



Pre-conditions	There is a catalogue of available datasets in the Marketplace. The Marketplace Participant has a login account as a Buyer and is able to login in the Marketplace.
Post-conditions	The Marketplace Participant has purchased the Dataset, and the transaction is logged in the system.
Basic Flow	<p>Step 1. The Marketplace Participant as Buyer searches in the catalogue for available datasets.</p> <p>Step 2. The Marketplace Participant as Buyer select a dataset and visualise related information (e.g., price and other data usage conditions).</p> <p>Step 3. The Marketplace Participant as Buyer confirms the dataset selected, accepts the purchase conditions and confirms the purchase.</p> <p>Step 4. The system moves the exact amount payable from the Marketplace Participant as Buyer to the Marketplace Participant as Seller.</p> <p>Step 5. The transaction is logged in the system.</p> <p>Step 6. The system notifies the Marketplace Participant as Seller about the sale.</p>
Alternate Flow(s)	The Marketplace Participant as Buyer processes the information but decides not to purchase any dataset or does not fulfil the required actions or dataset not found.
Exception Flow(s)	The Marketplace Participant as Buyer does not have sufficient funds; the transaction is cancelled.

3.11 (Marketplace Participant as Seller) Publish Data Services

Table 14 below describes the (Marketplace Participant as Seller) Publish Data Services UC.

Table 14: Publish Data Services Use Case

Use Case UC_DATASERVICE_1: Publish Data Services	
Brief Description	The Marketplace Participant as Seller inserts descriptive and technical information, including price and pricing scheme, about the offered Data Service in order to make it available on the Marketplace catalogue for selling.
Actor(s)	Marketplace Participant as Seller
Priority	High
Trigger	On demand.
Pre-conditions	The system is installed and active. The Marketplace Participant as Seller has a login account.
Post-conditions	Data Service is available on the Marketplace catalogue.



Basic Flow	Step 1. The Marketplace Participant as Seller accesses the area for publishing Data Services. Step 2. The Marketplace Participant as Seller provides the descriptive and technical information for the Data Service.
Alternate Flow(s)	
Exception Flow(s)	

3.12 (Marketplace Participant as Buyer) Purchase Data Services

Table 15 below describes the (Marketplace Participant as Buyer) Purchase Data Services UC.

Table 15: Purchase Data Services Use Case

Use Case UC_DATASERVICE_2: Purchase Data Services	
Brief Description	The Marketplace Participant as Buyer may like to purchase a Data Service. The Marketplace Participant as Buyer accesses the Marketplace catalogue and selects the Data Service for buying it. A notification is then sent to the Marketplace Participant as Seller, informing her/him of the sale.
Actor(s)	Marketplace Participant as Buyer, Marketplace Participant as Seller
Priority	High
Trigger	On demand.
Pre-conditions	The system is installed and active. The Marketplace Participant has a login account as a Buyer.
Post-conditions	Marketplace Participant has purchased the Data Service.
Basic Flow	Step 1. The Marketplace Participant logs onto the Marketplace. Step 2. The Marketplace Participant accesses the Marketplace catalogue. Step 3. The Marketplace Participant selects a Data Service for buying it. Step 4. The Marketplace Participant confirms the purchase of the Data Service. Step 5. The system moves the exact amount payable from the Marketplace Participant as Buyer to the Marketplace Participant as Seller. Step 6. The system logs the transaction. Step 7. The system notifies the Marketplace Participant as Seller about the sale.
Alternate Flow(s)	Data service not found
Exception Flow(s)	The Marketplace Participant as Buyer does not have sufficient funds; the transaction is cancelled.



3.13 (Marketplace Participant as Seller) Publish Charging Station Availability

Table 16 below describes the (Marketplace Participant as Seller) Publish Charging Station Availability UC.

Table 16: Publish Charging Station Availability Use Case

Use Case UC_CHARGINGSTATION_1: Publish Charging Station Availability	
Brief Description	The Marketplace Participant as Seller inserts descriptive information, including location, price and pricing scheme, about her/his charging station on the Marketplace catalogue for selling time slot availability.
Actor(s)	Marketplace Participant as Seller
Priority	Low
Trigger	On demand.
Pre-conditions	The system is installed and active. The Marketplace Participant as Seller has a login account.
Post-conditions	Charging station availability is published on the Marketplace catalogue.
Basic Flow	Step 1. The Marketplace Participant as Seller accesses the area for publishing Charging Station Availability. Step 2. The Marketplace Participant as Seller provides the descriptive information for her/his charging station and time slot availability. Step 3. The Marketplace Participant as Seller submits the information.
Alternate Flow(s)	
Exception Flow(s)	

3.14 (Marketplace Participant as Buyer) Purchase Charging Station Availability

Table 17 below describes the (Marketplace Participant as Buyer) Purchase Charging Station Availability UC.

Table 17: Purchase Charging Station Availability Use Case

Use Case UC_CHARGINGSTATION_2: Purchase Charging Station Availability	
Brief Description	The Marketplace Participant as Buyer may like to purchase availability of a charging station. The Marketplace Participant as Buyer accesses the Marketplace catalogue (or a map) and selects a charging station among the available ones. The Marketplace Participant as Buyer selects an available time slot for buying



	it. A notification is then sent to the Marketplace Participant as Seller, informing her/him of the sale.
Actor(s)	Marketplace Participant as Buyer, Marketplace Participant as Seller
Priority	Low
Trigger	On demand.
Pre-conditions	The system is installed and active. The Marketplace Participant as Buyer has a login account as a Buyer.
Post-conditions	Marketplace Participant as Buyer has purchased the charging station availability.
Basic Flow	<p>Step 1. The Marketplace Participant accesses the Marketplace catalogue.</p> <p>Step 2. The Marketplace Participant selects a charging station and an available time slot for buying it.</p> <p>Step 3. The Marketplace Participant confirms the purchase.</p> <p>Step 4. The system moves the exact amount payable from the Marketplace Participant as Buyer to the Marketplace Participant as Seller.</p> <p>Step 5. The system logs the transaction.</p> <p>Step 6. The system notifies the Marketplace Participant as Seller about the sale.</p>
Alternate Flow(s)	Charging station availability not found.
Exception Flow(s)	The Marketplace Participant as Buyer does not have sufficient funds; the transaction is cancelled.

3.15 (Marketplace Participant as Seller) Publish Apps

Table 18 below describes the (Marketplace Participant as Seller) Publish Apps UC.

Table 18: Publish Apps Use Case

Use Case UC_APP_1: Publish Apps	
Brief Description	The Marketplace Participant as Seller publishes a new data app in the app store.
Actor(s)	Marketplace Participant as Seller
Priority	High
Trigger	On demand
Pre-conditions	The App code is available, and the Marketplace Participant is authenticated via its login account. The Market Participant can login in the Marketplace.
Post-conditions	The app (optionally) is subjected to the certification workflow, is persisted in the data repository of the App Store and is listed as available in the overall App catalogue.



Basic Flow	<p>Step 1. The Marketplace Participant as Seller authenticates if a login session is expired or not available.</p> <p>Step 2. The Marketplace Participant as Seller activates the publish app function either directly via the App store interface or directly via its certified dataspace compliant connector.</p> <p>Step 3. The Marketplace Participant as Seller provides the required information that characterises the App being submitted, namely pricing.</p> <p>Step 4. The App Store validates the submission and makes it available, by setting its visibility state.</p> <p>Step 5. The Marketplace Participant as Seller verifies that the submitted App is available.</p>
Alternate Flow(s)	<p>Alternative flow if App certification is carried out:</p> <p>Step 1. The Marketplace Participant authenticates if a login session is expired or not available.</p> <p>Step 2. The Marketplace Participant activates the publish app function either directly via the App store interface or directly via its certified dataspace compliant connector.</p> <p>Step 3. The App Store fails the certification of the App and the App does not collect its certificate nor it is published in the App catalogue.</p>
Exception Flow(s)	<p>If App certification is enabled and the corresponding App to be published does not pass the certification process.</p>

3.16 (Marketplace Participant as Seller/Buyer) Purchase Apps

Table 19 below describes the (Marketplace Participant as Seller/Buyer) Purchase Apps UC.

Table 19: Purchase Apps Use Case

Use Case UC_APP_3: Purchase Apps	
Brief Description	Acquire Apps access usage rights.
Actor(s)	Marketplace Participant
Priority	Medium
Trigger	On demand
Pre-conditions	The App selected by the Marketplace Participant as Buyer is published and its visibility allows it to be browsed. The App requires payment. The Marketplace Participant as Buyer is authenticated via its login account.
Post-conditions	The selected App is transferred to the Marketplace Participant as Buyer.
Basic Flow	<p>Step 1. The Marketplace Participant browses the Apps available in the catalogue and selects the desired App.</p> <p>Step 2. The Marketplace Participant triggers the checkout process and pays the requested amount.</p>



	<p>Step 3. The Market validates the transaction and unlocks the transfer process of the App.</p> <p>Step 4. The Marketplace Participant collects the App.</p>
Alternate Flow(s)	
Exception Flow(s)	<p>The Marketplace Participant as Buyer does not have sufficient funds; the transaction is cancelled.</p> <p>The app becomes unavailable during the checkout transaction, reverting the payment.</p>

3.17 (Marketplace Participant as Energy Consumer) Modify Energy Contract Conditions

Table 20 below describes the (Marketplace Participant as Energy Consumer) Modify Energy Contract Conditions UC.

Table 20: Modify Energy Contract Conditions Use Case

Use Case UC_ENERGYCONTRACT_1: Modify Energy Contract Conditions	
Brief Description	The Marketplace Participant as Energy Consumer wants to modify the energy contract conditions with the Energy Trader.
Actor(s)	Marketplace Participant as Energy Trader, Marketplace Participant as Energy Consumer
Priority	High
Trigger	The Marketplace Participant as Energy Consumer interacts with ENERSHARE Marketplace to request available energy contract conditions. The request can be directed to a specific Energy Trader or to all available Energy Traders in the Marketplace.
Pre-conditions	The Marketplace Participant has a login account as an Energy Consumer and is able to login in the Marketplace. Energy related information is available in the Marketplace.
Post-conditions	The Marketplace Participant as Energy Consumer receives energy contract conditions and related Energy Trader contact information.
Basic Flow	<p>Step 1. (Optional) The Marketplace Participant as Energy Consumer publishes his/her energy consumption data.</p> <p>Step 2. The Marketplace Participant as Energy Consumer requests available energy contract conditions just to certain specific Energy Traders or to all Energy Traders in the data space.</p> <p>Step 3. The Marketplace Participant processes the information (no interaction with Marketplace).</p>



	Note: the possible negotiation of a new contract based on the conditions of step 3 is outside the scope of the Marketplace.
Alternate Flow(s)	
Exception Flow(s)	

3.18 (Marketplace Participant as Energy Community Promoter) Modify Distribution Coefficients in an Energy Community

Table 21 below describes the (Marketplace Participant as Energy Community Promoter) Modify Distribution Coefficients in an Energy Community UC.

Table 21: Modify Distribution Coefficients in an Energy Community Use Case

Use Case UC_COEFFICIENTSENERGYCOMMUNITY_1: Modify Distribution Coefficients in an Energy Community	
Brief Description	The Marketplace Participant as Energy Community Promoter wants to propose a modification of the distribution coefficients in an energy community (flexibility).
Actor(s)	Marketplace Participant as Energy Community Promoter, Marketplace Participant as Energy Community Approver
Priority	High
Trigger	The Marketplace Participant as Energy Community Promoter interacts with ENERSHARE Marketplace to propose a new distribution of the coefficients in the energy community.
Pre-conditions	The Marketplace filters the access to data depending on the Energy Community. The Marketplace Participant has a login account and is able to login in the Marketplace to the corresponding Energy Community. Information related with their community is available in the Marketplace.
Post-conditions	The new distribution of the coefficients in the energy community is agreed in the Marketplace. The transactions are logged in the system.
Basic Flow	Step 1. The Marketplace Participant as Energy Community Promoter requests a new distribution of the coefficients to the Energy Community. Step 2. The Marketplace Participants as Energy Community Approvers vote the new distribution and accept/reject the request. Step 3. The voting result is stored in the system.



	Note: If applicable, the communication of the new distribution of coefficients to the DSO is carried out outside the scope of the Marketplace.
Alternate Flow(s)	
Exception Flow(s)	

3.19 (Marketplace Participant as Auction Promoter) Publish Cross-domain Services

Table 22 below describes the (Marketplace Participant as Auction Promoter) Publish Cross-domain Services UC.

Table 22: Publish Cross-domain Services Use Case

Use Case UC_CROSSDOMAINSERVICE_1: Publish Cross-domain Services	
Brief Description	The Marketplace Participant as Auction Promoter accesses the auction area and inserts metadata about the service so that it becomes available in a catalogue of active auctions. The metadata contains descriptive/technical information about the service. The Marketplace Participant as Auction Promoter sets up start and stop of auction. The system starts the auction.
Actor(s)	Marketplace Participant as Auction Promoter
Priority	Medium
Trigger	On demand
Pre-conditions	The system is installed and active. The Marketplace Participant as Auction Promoter has a login account.
Post-conditions	The cross-domain service is available in the auction catalogue.
Basic Flow	<p>Step 1. The Marketplace Participant as Auction Promoter accesses the area for setting up an auction for exchanging a service.</p> <p>Step 2. The Marketplace Participant as Auction Promoter provides the descriptive/technical information for the service.</p> <p>Step 3. The Marketplace Participant as Auction Promoter sets up start and stop of the auction.</p> <p>Step 4. The system starts the auction.</p>
Alternate Flow(s)	
Exception Flow(s)	



3.20 (Marketplace Participant as Auction Bidder) Propose exchange to obtain a Cross-domain Service

Table 23 below describes the (Marketplace Participant as Auction Bidder) Propose exchange to obtain a Cross-domain Service UC.

Table 23: Propose exchange to obtain a Cross-domain Service Use Case

Use Case UC_CROSSDOMAINSERVICE_2: Propose exchange to obtain a Cross-domain Service	
Brief Description	While the auction is active, the Marketplace Participant as Auction Bidder selects the cross-domain service from the active auctions list. The Marketplace Participant as Auction Bidder proposes something to be provided in exchange, providing descriptive- /technical information.
Actor(s)	Marketplace Participant as Auction Bidder
Priority	Medium
Trigger	On demand
Pre-conditions	The system is installed and active. The Marketplace Participant has a login account as an Auction Bidder.
Post-conditions	The Marketplace Participant as Auction Bidder has proposed something (dataset, service, etc.) in exchange with a cross-domain service.
Basic Flow	Step 1. The Marketplace Participant as Auction Bidder accesses the active auctions catalogue. Step 2. The Marketplace Participant as Auction Bidder selects a cross-domain service. Step 3. The Marketplace Participant as Auction Bidder propose an exchange, providing descriptive/technical information.
Alternate Flow(s)	
Exception Flow(s)	

3.21 (Marketplace Participant as Auction Promoter) Selects best offer as exchange for a Cross-domain Service

Table 24 below describes the (Marketplace Participant as Auction Promoter) Selects best offer as exchange for a Cross-domain Service UC.

Table 24: Selects best offer as exchange for a Cross-domain Service Use Case

Use Case UC_CROSSDOMAINSERVICE_3: Selects best offer as exchange for a Cross-domain Service



Brief Description	After the closure of the auction, the Marketplace Participant as Auction Promoter accesses the auction area and selects the best offer received. The Marketplace Participant as Auction Bidder is notified that the exchange is accepted.
Actor(s)	Marketplace Participant as Auction Promoter, Marketplace Participant as Auction Bidder
Priority	Medium
Trigger	On demand
Pre-conditions	The system is installed and active. The Marketplace Participant as Auction Promoter has a login account.
Post-conditions	The Marketplace Participant as Auction Promoter has selected the best offer (dataset, service, etc.) in exchange with her/his cross-domain service. The Marketplace Participant as Action Bidder has been notified that his/her offer has been selected.
Basic Flow	Step 1. The Marketplace Participant as Auction Promoter accesses the service auction area. Step 2. The Marketplace Participant as Auction Promoter visualises the offers proposed and selects the best offer. Step 3. The system logs the transaction. Step 4. The system notifies the Marketplace Participant as Auction Bidder by email about the accepted exchange.
Alternate Flow(s)	
Exception Flow(s)	

3.22 (Marketplace Participant as Auction Bidder) Access Cross-domain Service auction result

Table 25 below describes the (Marketplace Participant as Auction Bidder) Access Cross-domain Service auction result UC.

Table 25: Access Cross-domain Service auction result Use Case

Use Case UC_CROSSDOMAINSERVICE_4: Access Cross-domain Service auction result	
Brief Description	Marketplace Participant as Auction Bidder accesses her/his offer result.
Actor(s)	Marketplace Participant as Auction Bidder
Priority	Medium
Trigger	On demand.
Pre-conditions	The system is installed and active. The Marketplace Participant as Auction Bidder has a login account. The Marketplace Participant as Auction Promoter has selected best auction offer.
Post-conditions	Marketplace Participants as Auction Bidder are aware about auction result.



Basic Flow	Step 1. The Marketplace Participant as Action Bidder accesses the action result.
Alternate Flow(s)	
Exception Flow(s)	

3.23 (Marketplace Participant as Barter Data Producer/Consumer) Exchange Data/Collaborative Data Analytics Services

Table 26 below describes the (Marketplace Participant as Barter Data Producer/Consumer) Exchange Data/Collaborative Data Analytics Services UC.

Table 26: Exchange Data/Collaborative Data Analytics Services Use Case

Use Case UC_EXCHANGE_1: Exchange Data/Collaborative Data Analytics Services	
Brief Description	Data Exchange flows are established between data or data analytics services between data producers and data consumers. This use case covers scenarios where valuable data for a specific service (e.g., load and renewable energy time series forecasting) is distributed across multiple owners/devices and monetary and non-monetary (barter) incentive mechanisms are needed to foster data sharing and enable collaborative data analytics.
Actor(s)	Marketplace Participant as Barter Data Producer, Marketplace Participant as Barter Data Consumer, Marketplace Participant as Barter Operator
Priority	High
Trigger	On demand, for each data exchange request; schedule data exchange (e.g., daily)
Pre-conditions	Consumers and Producers have a deployment for a certified connector or are registered in the Market and hold a valid identity to operate the data space or the Market. A market session is established.
Post-conditions	Data Exchange is unlocked, data flow between data producer connector is established with the data consumer connector. Data analytics output is shared with the Barter Data/Service Producer/Consumer.
Basic Flow	<p>Step 1 (Barter Data Consumer): Submit requests (necessary data, and maximum price – optional) for the data sets from the available catalogue.</p> <p>Step 2 (Barter Data Producer): Agrees with terms and conditions for data provision and submits offers (data, and minimum price – optional).</p> <p>Step 3 (Barter Operator): Receives the data access/provision requests and validates them.</p> <p>Step 4 (Barter Operator): Clears the market (incentive mechanism) and defines data exchange (and set a fair payment distribution in case of data monetization).</p> <p>Step 5 (Barter Data Producer): Receives the funds in case of payment.</p>



	Step 6 (Barter Data Consumer): Receives the data resources and undergoes any checkout process to pay for the data exchange.
Alternate Flow(s)	In case of service provision, instead of raw data provision from the Barter Data Producer: Step 1 (Barter Data Consumer): Submit requests (offer) for the data-centric services from the available catalogue. Step 2 (Barter Data Producer): Agrees with terms and conditions for data provision and submits offers (data, and minimum price – optional). Step 3 (Barter Operator): Receives the data access/service provision requests and validates them. Step 4 (Barter Operator): Clears the market (incentive mechanism) and defines data exchange (and fair payments in case of data monetization). The final data for the Data Consumer is the output of a contracted data analytics service, instead of raw data from the Data Producer. Step 5 (Barter Data Producer): Receives the funds in case of payment. Step 6 (Barter Data Consumer): Receives the analytical resources.
Exception Flow(s)	Actor depicted embody automate the processes described.

3.24 (Marketplace Participant as Buyer) Rate Asset/Seller

Table 27 below describes the (Marketplace Participant as Buyer) Rate Asset/Seller UC.

Table 27: Rate Asset/Seller Use Case

Use Case UC_REPUTATION_1: Rate Asset/Seller	
Brief Description	Marketplace Participant as Buyer wants to rate an asset or a Seller.
Actor(s)	Marketplace Participant as Buyer, Market Participant as Seller
Priority	Low
Trigger	On demand.
Pre-conditions	The system is installed and active. The Marketplace Participant as Buyer has a login account. The Marketplace Participant as Buyer has purchased an asset.
Post-conditions	Marketplace Participants as Buyer has rated asset/Seller.
Basic Flow	Step 1. The Marketplace Participant as Buyer accesses the transaction history area. Step 2. The Marketplace Participant as Buyer rates the purchased asset and/or the seller. Step 3. The system notifies the Market Participant as Seller about the rating.
Alternate Flow(s)	



Exception Flow(s)	
--------------------------	--

3.25 (Marketplace Participant as Seller) Visualise Ratings

Table 28 below describes the (Marketplace Participant as Seller) Visualise Ratings UC.

Table 28: Visualise Ratings Use Case

Use Case UC_REPUTATION_2: Visualise Ratings	
Brief Description	Marketplace Participant as Seller wants to know a rating for himself or the asset sold.
Actor(s)	Marketplace Participant as Seller
Priority	Low
Trigger	On demand.
Pre-conditions	The system is installed and active. The Marketplace Participant as Seller has a login account. The Marketplace Participant as Seller has sold assets.
Post-conditions	Marketplace Participants as Seller knows his ratings.
Basic Flow	Step 1. The Marketplace Participant as Seller accesses the transaction history area. Step 2. The Marketplace Participant as Buyer visualise ratings.
Alternate Flow(s)	
Exception Flow(s)	



4 Requirements

The following table lists the functional requirements of the ENERSHARE Marketplace.

Table 29: Functional requirements definition

ReqID	Name	Priority	Derived from	Description
FR_1	Registration	High	UC_MANAGEMENT_1	The system shall allow Marketplace Participants to register to the Marketplace as Sellers, Consumers or both.
FR_2	Profile update	Medium	UC_MANAGEMENT_5	The system should allow Marketplace Participants to update their personal profiles.
FR_3	Delete registration	Medium	UC_MANAGEMENT_7	The system should allow Marketplace Participants to delete their personal profiles from the Marketplace.
FR_4	Login	High	UC_MANAGEMENT_2	The system shall allow Marketplace Participants to log into the Marketplace.
FR_5	Logout	High	UC_MANAGEMENT_3	The system shall allow Marketplace Participants to logout from the Marketplace.
FR_6	Profile information storage	High	UC_MANAGEMENT_1, UC_MANAGEMENT_2, UC_MANAGEMENT_3, UC_MANAGEMENT_4, UC_MANAGEMENT_5, UC_MANAGEMENT_6, UC_MANAGEMENT_7	The system shall store Marketplace Participants' profiles and logon details.
FR_7	View profile	High	UC_MANAGEMENT_4, UC_MANAGEMENT_5,	The system shall be able to display stored data related to Marketplace Participants.





			UC_MANAGEMENT_6, UC_MANAGEMENT_7	
FR_8	Validate registrations	High	UC_MANAGEMENT_4	The system shall allow the Marketplace Administrator as Operator to validate the Participant's registration.
FR_9	Validate profile updates	Medium	UC_MANAGEMENT_6	The system should allow the Marketplace Administrator as Operator to validate updates in the Marketplace Participants' profiles.
FR_10	Validation notifications	Medium	UC_MANAGEMENT_4, UC_MANAGEMENT_6	The system should notify Marketplace Participants of the validation process result.
FR_11	Delete registration	Medium	UC_MANAGEMENT_7	The system should allow Marketplace Participants to delete their registration.
FR_12	Catalogue availability	High	UC_DATASET_1, UC_DATASET_2, UC_DATASERVICE_1, UC_DATASERVICE_2, UC_CHARGINGSTATION_1, UC_CHARGINGSTATION_2, UC_APP_1, UC_APP_2, UC_APP_3, UC_EXCHANGE_1	The Marketplace shall provide a catalogue with information about available datasets, services and apps, charging station, including descriptions, attributes, usage rights, and associated metadata.
FR_13	Search and filtering in the catalogue	High	UC_DATASET_2, UC_DATASERVICE_2, UC_CHARGINGSTATION_2,	The Marketplace shall offer intuitive search capabilities and filters to facilitate exploration and retrieval of metadata.





			UC_EXCHANGE_1	
FR_14	Publish Data Services	High	UC_DATASERVICE_1, UC_EXCHANGE_1, UC_APP_1	The system shall allow the Marketplace Participants as Seller to publish Data Services for selling.
FR_15	Purchase Data Services	High	UC_DATASERVICE_2, UC_EXCHANGE_1, UC_APP_3	The system shall allow the Marketplace Participants as Consumer to purchase Data Services.
FR_16	Publish Dataset	High	UC_DATASET_1, UC_EXCHANGE_1, UC_APP_1	The system shall allow the Marketplace Participants as Seller to publish Data for selling.
FR_17	Purchase Dataset	High	UC_DATASET_2, UC_EXCHANGE_1, UC_APP_3	The system shall allow the Marketplace Participants as Consumer to purchase Data.
FR_18	Publish Charging Station Availability	Low	UC_CHARGINGSTATION_1	The system may allow the Marketplace Participants as Seller to publish charging station availability data for selling.
FR_19	Purchase Charging Station Availability	Low	UC_CHARGINGSTATION_2	The system may allow the Marketplace Participants as Buyer to purchase charging station availability time slots.
FR_20	Select Charging Stations from map	Low	UC_CHARGINGSTATION_2	The system may allow the Marketplace Participants as Buyer to view charging station locations on a map and select the available ones.
FR_21	Select Charging Stations from catalogue	Low	UC_CHARGINGSTATION_2	The system may allow the Marketplace Participants as Buyer to view charging station locations in a catalogue and select the available ones.





FR_22	Schedule cross-domain service auction	Medium	UC_CROSSDOMAINSERVICE_1	The system should allow Marketplace Participants as Auction Promoters to publish a service and schedule an auction for cross-domain exchange.
FR_23	Store cross-domain service auction information	Medium	UC_CROSSDOMAINSERVICE_1, UC_CROSSDOMAINSERVICE_2, UC_CROSSDOMAINSERVICE_3, UC_CROSSDOMAINSERVICE_4	The system should record information related to cross-domain auctions.
FR_24	Propose cross-domain service auction	Medium	UC_CROSSDOMAINSERVICE_2	When the auction is running, the system should allow Marketplace Participants as Auction Bidders to propose an exchange (dataset, data service, etc.).
FR_25	Select best proposal for cross-domain service auction	Medium	UC_CROSSDOMAINSERVICE_3	When the auction is stopped, the system should allow Marketplace Participants as Auction Promoters to select the best proposal for exchange (dataset, data service, etc.).
FR_26	Auction result notification	Medium	UC_CROSSDOMAINSERVICE_3	The system should notify by email Marketplace Participants as Auction Bidders about auction results.
FR_27	Access auction result notification	Medium	UC_CROSSDOMAINSERVICE_4	The system should allow Marketplace Participants as Auction Bidders to visualise auction results.
FR_28	Transaction storage	High	UC_TRANSACTION_1	The system shall store Marketplace Participants' transaction details.
FR_29	Transaction history	High	UC_TRANSACTION_1	The system shall be able to display transaction history to Marketplace Participants.
FR_30	Log transactions	High	UC_TRANSACTION_1	The Marketplace shall provide a secure mechanism to log transactions.



FR_31	Trust and reputation management	Low	UC_REPUTATION_1, UC_REPUTATION_2	The Marketplace may implement trust and reputation mechanisms to rate apps, services, datasets, and participants.
FR_32	Payment support	Medium	UC_DATASET_2, UC_DATASERVICE_2, UC_CHARGINGSTATION_2, UC_APP_3	The Marketplace should support payment of apps, services or datasets.
FR_33	Sale notifications	High	UC_DATASET_2, UC_DATASERVICE_2, UC_CHARGINGSTATION_2, UC_APP_3	The system shall notify Marketplace Participants as Seller of the sale of their Data Services, Datasets, Apps and Charging Station Availabilities.
FR_34	Notification of data operation call to the clearing House	High	UC_TRANSACTION_1	Upon data consumer request for data, a notification shall be sent to the Clearing House for logging the data operation request.
FR_35	Notification of data operation call reception to the Clearing House	High	UC_DATASET_1, UC_DATASERVICE_1, UC_CHARGINGSTATION_1, UC_APP_3, UC_CROSSDOMAINSERVICE_3	Upon Data Provider reception of data consumer's request, a notification shall be sent to the Clearing House for logging reception.
FR_36	Clearing house logs all transactions in a persistent data storage	High	UC_TRANSACTION_1	The Clearing House shall log all transactions in a persistent data storage, ensuring data provenance and traceability.
FR_38	Notification of data operation result sent to the Clearing House	High	UC_DATASET_1, UC_DATASERVICE_1, UC_CHARGINGSTATION_1, UC_APP_3, UC_CROSSDOMAINSERVICE_3	Notification of data operation result shall be sent to clearing House by the Data Provider.
FR_39	Notification of data operation result received at the Clearing House	High	UC_DATASET_2, UC_DATASERVICE_2, UC_CHARGINGSTATION_2, UC_APP_3, UC_CROSSDOMAINSERVICE_3	Notification of data operation shall result received at Clearing House.
FR_40	Auditing and tracking	Low	UC_TRANSACTION_1	The system may allow auditing and tracking of data transactions for determining accountability and resolving possible conflicts.





FR_41	Monetization of all exchanged data with fairness	High	UC_EXCHANGE_1	Data Producers and Data Consumers shall give a settlement price for the data or service they provided to the Data Monetization and Barter Sharing Incentive Module. Payment division among Data Producers should be fair (e.g., according to data value).
FR_42	Possibility to exchange data-by-data (non-economic data exchange)	High	UC_EXCHANGE_1	Data Producers and Data Consumers shall be able to exchange data without financial compensations and solely based in data properties (e.g., information content).
FR_43	Two-Factor Authentication	Medium	UC_EXCHANGE_1	The system should provide the option for users to enable two-factor authentication for enhanced security during login.
FR_44	Market Analytics and Insights	Medium	UC_EXCHANGE_1	The system should provide analytics and insights on market trends, transaction history, and user behaviour to assist Participants in decision-making
FR_45	Forgot Password Recovery	High	UC_EXCHANGE_1	The system shall allow users to recover their password through a secure process that includes sending a password reset link to their registered email.
FR_46	Integrated Support System	Medium	UC_EXCHANGE_1	The system should include an integrated support system, allowing users to submit queries, report issues, access FAQ's and receive assistance from administrators.
FR_47	Vote for different coefficients distribution proposals	Medium	UC_COEFFICIENTSENERGYCOMMUNITY_1	The system should provide the means to carry out a voting procedure for a coefficient distribution





				proposal in an Energy Community.
FR_48	Manage several Energy Communities	High	UC_COEFFICIENTSENERGYCOMMUNITY_1	The system shall support data privacy and filter data according to each Energy Community.
FR_50	Publish data with a specific profile/user	High	UC_ENERGYCONTRACT_1	The system shall support publishing data with specific profiles/users.



5 Static logical view of the Architecture

The following Figure 3 depicts the ENERSHARE Marketplace architecture. In the following, the main components are described.

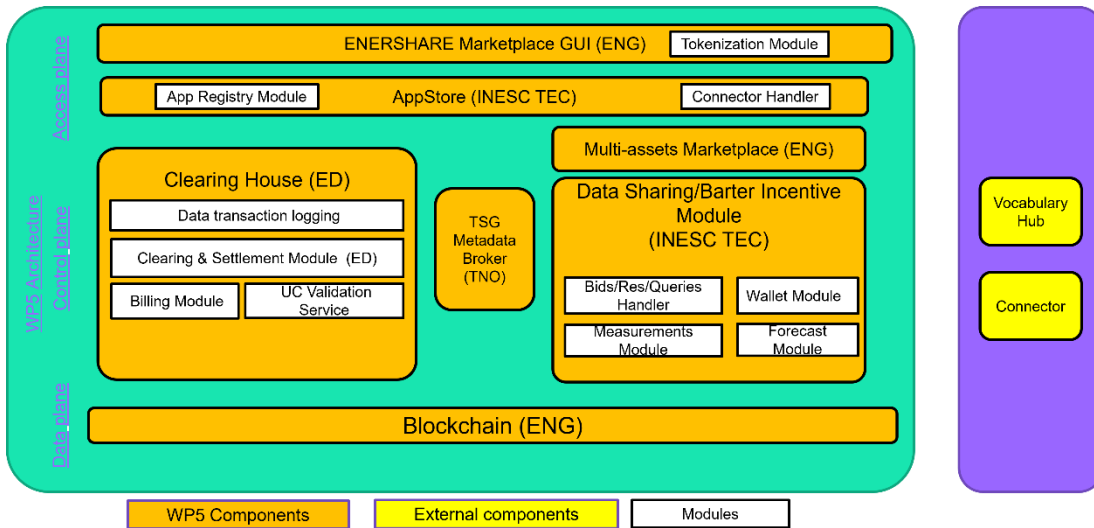


Figure 3: ENERSHARE Marketplace Architecture

5.1 ENERSHARE Marketplace Graphical User Interface (GUI)

The Marketplace GUI should provide the access to all the functionalities developed within WP5, or maybe in other WPs as well, as a common entry point, for instance the access to the AppStore. To easily interact with the Marketplace, the GUI should provide the access to all the main functionalities of the system, directly from the main menu (e.g., a left/right side menu). The Marketplace GUI should provide register, login and logout functionalities for the Marketplace Participant and Marketplace Administrator users.

A Marketplace Participant user should have access to different sections for searching different types of assets, using metadata to filter the results. A Marketplace Participant as a Seller can decide to sell an asset. So, after going to the related section of the Marketplace, the user shall be able to provide all the information required for publication of the asset in the Marketplace, defining, for example, the name, the categories, the description, and the price of the asset. The Marketplace Participant as a Buyer can buy an asset. So, from the menu, the Marketplace Participant can select a specific type of asset and search for the one he or she would like to



purchase or to access. Both Marketplace Participants, Seller and Buyer, may choose to participate in different types of market sessions.

Finally, a Marketplace Participant should be able to modify all its personal information directly from a specific section of the GUI.

A Marketplace Administrator as Operator logged into the Marketplace GUI should be able to manage all other users (Marketplace Participants) within the Marketplace platform, directly from the Marketplace GUI application, by having a different, private section where she or he can view all users participating in the Marketplace, with the ability to search and filter results. The Marketplace Administrator as Operator should also view the list of registration requests and decide whether to accept them or not, in accordance with the established acceptance/rejection criteria.

The Marketplace should have a modern and responsive GUI that facilitates the interaction with all the functionalities offered by the WP5 components and optimize usability and the learning curve in using it.

5.2 AppStore

The App Store integrates into the International Data Spaces (IDS) ecosystem as one of the building blocks. It interfaces with the IDS connector and enables Data Apps to be distributed within the data space. An IDS App is a reusable asset that can be downloaded from the App Store and can be deployed and monitored by an IDS Connector. As Data Apps are operated and executed in a secure environment, they are made available as virtualized containers.

Data Apps are seen in the IDS as independent, functional, and re-usable assets that can be deployed, executed, and operated on a connector and its underlying data or services.

Participants use the App Store to browse the available Apps, verifying their requirements and functionalities before downloading them. When users download Data Apps, they are instantiated in the Participant's IDS Connector instance. Participants also use the App Store to publish their own Data Apps so they can be used by others. Moreover, Participants use the App Store to prototype fast new Data Apps, focusing on the business-related data acquisition or data transformations, while ensuring key data acquisition and integration with the IDS Connector environment is taken care of.

Data Apps are available in two different categories, namely: system adapters and smart data apps. The apps that fall under the category of *system adapters* have the task of connecting all types of data sources or data sinks and making them available to the underlying Connector. This includes, for example, connections to databases, web services or Big Data systems. Furthermore, system adapters can be used to transform data models from connected data



sources into a standardized format. In addition, system adapters can be used to enrich connected data with additional metadata.

Unlike system adapters, the category of *smart data* apps includes all applications that manipulate the data flow within a connector in some way. The main tasks of a smart data app include processing or transforming data. Examples include applications for data quality assurance or machine learning. The data that a smart data app processes and makes available again are usually already enriched with metadata.

The overall operation of the App Store can be illustrated according to the workflow depicted in Figure 4, establishing key intermediate steps. App certification is considered as an optional cycle within the workflow, whose specific terms will be off set to a later release of this component.

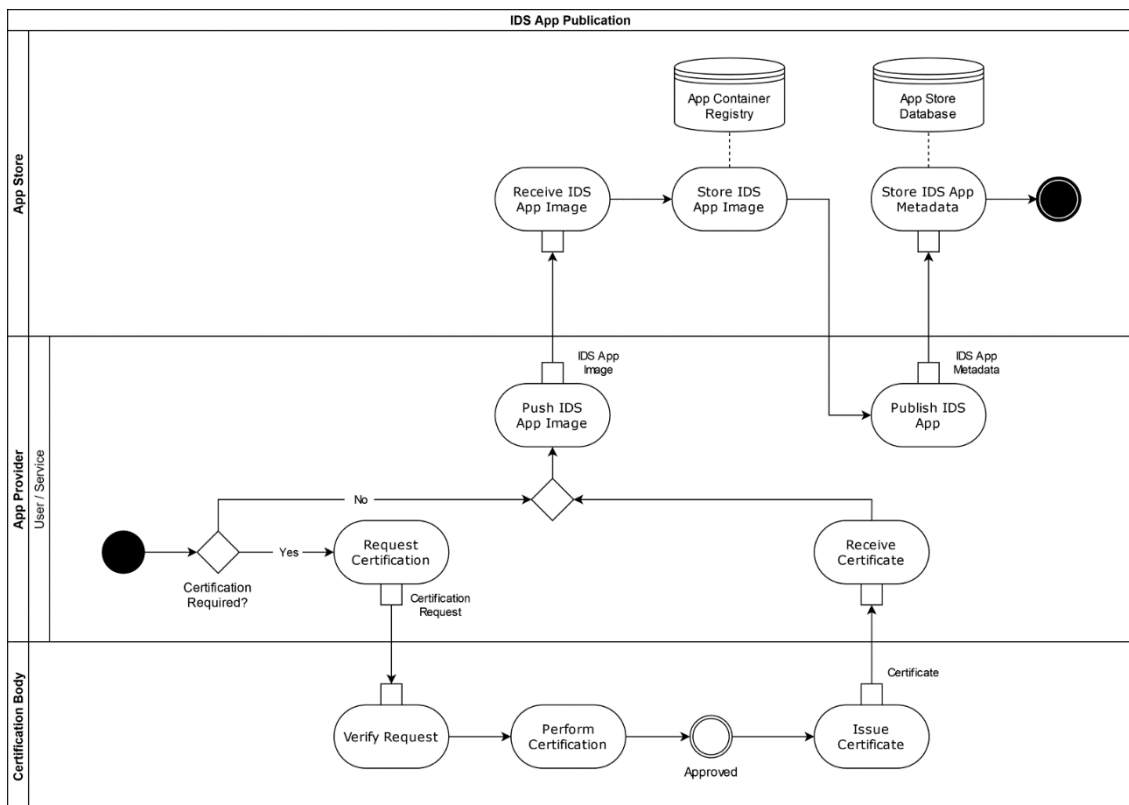


Figure 4: App Store lifecycle workflow

The key functionalities of the App Store allow it to:

- **Publish Data Apps.**

The owner/integrator/service provider of the App uses the App Store to publish it within the ecosystem. Each App is characterised belonging to one of two categories: *system adapters* – used to integrate service provider’s data sources with the connector; *smart*





data apps – that manipulate data either before or after a data exchange transaction occurs. Each data app type will render distinct metadata requirements to capture the characteristics of the App. The App is *registered* within the App Store repository, and its container image is uploaded to the App Store container image repository.

- **Browse available Data Apps.**
Users of the App Store browse the available Data Apps, considering filtering and searching tools on the Data Apps metadata. Possible Data Apps categorisation may be in place to assist the searching process.
- **Retrieve Data App.**
The user locates the App to be collected via browse available data apps step. The App is downloaded to the user side and instantiated within the user's IDS compliant Connector.
- **Manage information and metadata about Data Apps.**
The App Store considers a data store to locally hold information and metadata that characterizes each of the available Data Apps. The data managed by the App Store does not consider the App operational data, rather the characterising metadata. The App Store makes available services that relay metadata from available Data Apps to assist to ***Browse available Data Apps***.
- **Certification of Data Apps.**
The App Store considers the liaison with a certification body to subject data apps to a series of tests that ensure they conform with IDSA. Data Apps that that are certified get highlighted. Uncertified apps are still considered, but without the highlight.

Section 6.1 details the architecture and provides an overview of several implementation details.

5.3 Clearing House

As a dataspace component, the IDS Clearing House (CH)⁹ is an important intermediary in the whole IDS ecosystem. Basically, the IDS Clearing House mediates between a Data Provider (DP) and a Data Consumer (DC), making sure both parties meet their contractual obligations, such as:

⁹ https://internationaldataspaces.org/wp-content/uploads/dlm_uploads/IDSA-White-Paper-Specification-IDS-Clearing-House-.pdf



- the data sharing process from the Data Producer to the Data Consumer according to defined Usage Contracts and Data Usage Policies
- the use of data from the Data Consumer according Usage Contracts and Data Usage Policies along with the agreed payment to the Data Produces

For each data exchange transaction, the data producer attaches metadata to the data requested by the Data Consumer, specifying e.g., data usage restrictions, pricing information, payment entitlement, time of validity, etc. This way, the data producer can specify a Data Usage Policy as deemed appropriate, making sure data sovereignty is guaranteed. The IDS Clearing House provides two basic functions for all financial and data exchange transactions taking place in the IDS ecosystem: (1) clearing and (2) settlement on the basis of transaction logging. In IDS, the activities carried out by the IDS Clearing House are separated from other services, since these activities are executed at different stages of the lifecycle of the data-sharing support processes. The IDS Clearing House interfaces with, and relies upon, the support services described above in order to provide the functions for managing data-sharing and payment processes after mutual agreement between the Data Provider and the Data Consumer (i.e., managing actual data-sharing transactions in accordance with data-sharing agreements including logging and reporting thereof). It plays an important role in providing legal, financial and technical support functions; i.e., functions for clearing (prior to a data exchange transaction) and for settlement (after a data exchange transaction).



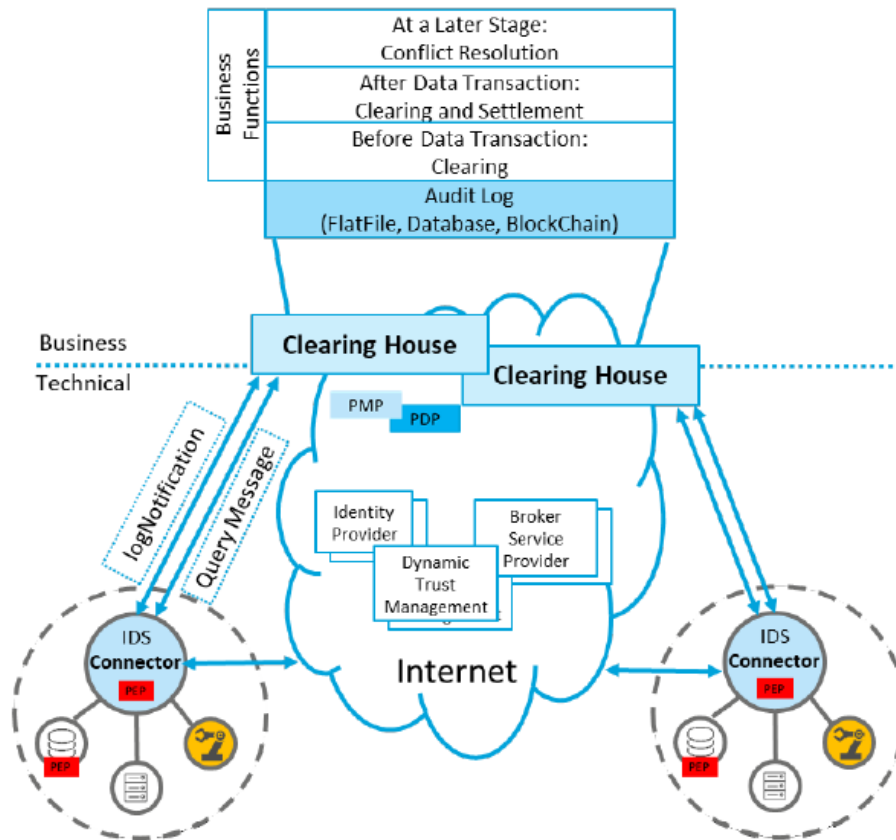


Figure 5: Logic and functions of the IDS Clearing House

Figure 5 describes the logic of clearing house intervention along with a complete list of the generic business functions that shall be covered. The analysis of such business functions containing the complete set of Clearing House features according to literature is provided below:

IDS Connector Functionalities:

1. Prior to sharing data: clearing functions

Clearing of data-sharing transaction:

- Legal: Verifying Usage Contract and Data Usage Policy
- Financial: Verifying payment conditions
- Technical: Enabling execution of transaction and binding transaction to an instance of a data-sharing agreement and Usage Contract

2. During the data-sharing process: monitoring and logging functions

Settlement functions (prior to and during data-sharing process):



- Discharging of data-sharing transaction
- Logging of transaction metadata
- Tracing data provenance
- Monitoring and reporting of data transactions
- Auditing and tracking of data transactions for determining accountability and resolving possible conflicts
- Billing and invoicing of data transactions

3. After sharing data: settlement functions

Settlement functions (after sharing data, or in case of not sharing any data) for conflict resolution:

- Investigating claim on violation of Usage Contract and/or Data Usage Policy
- Enforcing action upon violation of Usage Contract and/or Data Usage Policy
 - Legal: Escalate to a court
 - Technical: Block a participant via Identity Provider or downgrade its degree of trust using DTM
 - Financial: Request financial compensation

Within the context of ENERSHARE, the foremost importance of the Clearing House relies on the insurance of data provenance and thus providing the necessary logging and transcription features.

Particularly, the Dataspace Connector has local logging, that can be changed following intrinsic configuration steps. In addition, it logs some information to the Clearing House:

- finalized contract agreements,
- data usage (if noted in a usage policy),
- incoming and outgoing ArtifactRequestMessages,
- and incoming and outgoing ArtifactResponseMessages.

The Clearing House stores the contact agreements, information about data request, data response and data usage made under the corresponding agreement. Such information can be queried from all connectors that are part of the contract agreement.



5.4 Metadata Broker

According to the IDS-RAM 4, each connector in a data space has its own catalogue functionality that contains its resource self-descriptions. A Connector may offer multiple resources and each connector’s catalogue may therefore contain many resource self-descriptions. For a Consumer to find specific resources across different consumers requires querying of multiple connectors, i.e., multiple catalogues, and then integrating the results.

The Metadata Broker is a component that acts as a centralized catalogue of resource self-descriptions in a data space¹⁰. Or, more specific, as a central access point to resource self-descriptions across Connectors. Providers can register themselves and their services at the Metadata Broker so that they become discoverable by Consumers^{11 12}.

The Metadata Broker contains an endpoint for the registration, publication, maintenance, and query of self-descriptions. This endpoint is used by Providers to create and manage self-descriptions and by consumers to discover them.

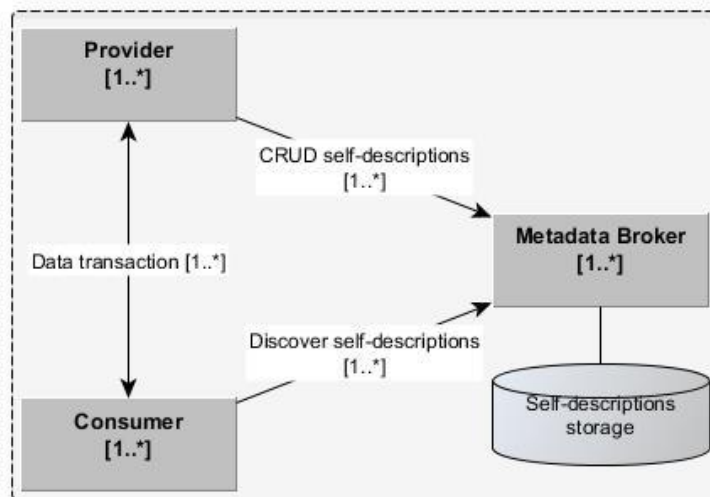


Figure 6: Logical view of the Metadata Broker

¹⁰ Gaia-X Federation Services / Federated Catalogue: https://docs.gaia-x.eu/technical-committee/architecture-document/latest/federation_service/#federated-catalogue

¹¹ IDS-RAM 4: https://docs.internationaldataspaces.org/ids-ram-4/layers-of-the-reference-architecture-model/3-layers-of-the-reference-architecture-model/3_5_0_system_layer/3_5_4_metadata_broker

¹² Note that the Metadata Broker is -in fact- a connector itself. Its service is exposure and management of self-descriptions across Connectors.

The Metadata Broker plays an intermediary role during orchestration of data transactions between providers and consumers, illustrated in Figure 6. A Provider can register, publish, and update self-descriptions in the Metadata Broker, which translates as functionalities to create, update, and delete (CRUD) self-descriptions. After publication, the self-descriptions can be discovered by Consumers who query the Metadata Broker. The self-descriptions will contain the required information for the Consumer to find best-matching services and connect to the provider to consume the resource. The Consumer will then connect to the Provider for the actual data transaction.

The Metadata Broker only acts as an intermediate publication- and discovery service for self-descriptions. Once the Consumer has received the needed information about a Provider the Metadata Broker no longer plays a role in the data interactions between Providers and Consumers.

5.5 Multi-assets Marketplace

The Multi-assets Marketplace module should include all the logical functions of the ENERSHARE Marketplace or could interact with external modules to use specific functions not implemented within the Marketplace itself. This module exposes all the functionality to buy and sell access to all the different assets provided by WP5, such as data, data services, energy, cross-domain services. In addition, it should provide access to the AppStore for buying and selling applications and provide the ability to participate in a different type of Marketplaces with bids and sessions.

The Multi-assets Marketplace module should interact with the Metadata Broker to provide new assets published by a Marketplace Participant and to access all catalogued assets to be returned to the Marketplace Participant who is searching for assets to purchase, providing asset search and filter capabilities.

The Multi-assets Marketplace should interact with the Clearing House, to manage the exchange activities during the selling of an asset. All the transactions should be managed invoking the clearing, monitoring and settlement functionalities of the Clearing House.

Before the exchange of assets between the Marketplace Participants, the Multi-assets Marketplace should check that all the preliminary requirements are met, through the usage of the corresponding functionalities in the Clearing House. After the preliminary stage, during the exchange of an asset it should use the monitoring capabilities of the Clearing House and store all the metadata related to the transaction, directly or using the Clearing House, inside the Blockchain (see section 5.7) to persist all the information required. Finally, after the exchange of assets, the Marketplace should provide the functionalities to check that all the usage restrictions of a sold asset are met using the settlement functionalities of the Clearing House.



5.6 Data Monetization and Barter Sharing Incentive Module

Clean and reliable energy sources are becoming increasingly vital for power generation globally. Solar and wind farms are gaining prominence as significant contributors to the electricity grid. Precise forecasting of energy production from these renewable sources is crucial for efficient energy management and ensuring grid stability. However, accurate forecasting poses significant challenges due to the inherent variability and uncertainty associated with renewable energy sources.

To address these challenges, we have developed a Data Marketplace with collaborative forecasting capabilities where our data Marketplace aims to promote cooperation among various Data Owners and improve the overall quality of energy forecasting. By leveraging advanced machine learning algorithms, our platform facilitates the buying and selling of energy production data indirectly through the application and computation of collaborative forecasting services.

In addition to facilitating data exchange, our Marketplace introduces a critical factor - the ability to evaluate the value of the data provided by a producer. We have implemented mechanisms to assess the relevance and quality of the data contributed by data producers (sellers), allowing for market valuation of the data considering its contribution for the overall forecast accuracy.

This section provides an overview of the Predico Data Marketplace objectives, data types implemented, communication methodology, data contribution techniques, future data integration strategies, software, and underlying technology.

[Note: Detailed information of data market functions is provided in the Appendix section of this document.]

5.6.1 Data Contribution Methods

Predico offers both manual and automated data contribution methods. Users can choose for the manual input through a dedicated Desktop Application or automate data transfers using APIs allowing consistent updates, augmenting the accuracy of forecasts by providing up-to-date information.

5.6.2 Data Types

The Predico Data Marketplace is designed to accommodate an array of energy-related datasets. These datasets encompass solar and wind energy production data, forming the base for reliable forecasting and informed energy management decisions. Furthermore, the platform's flexibility allows seamless integration of future data types such as solar irradiance, wind speed, and temperature.



5.6.3 Market Functions

The platform functionalities are geared towards a seamless user experience:

- **Registration with Cryptocurrency Wallet:** Users can register with the Marketplace while instantly is created a secure cryptocurrency wallet. This wallet becomes an essential tool for secure transactions within the Marketplace.
- **Authentication Mechanisms:** Rigorous authentication protocols guarantee secure user access, maintaining platform integrity.
- **Portfolio Management:** Users can actively manage their data resources within the platform. This encompasses data creation, organization, and performance monitoring, allowing users to optimize their contributions.

5.6.4 Collaborative Forecasting

The Market Engine collaborative forecasting function combines data from multiple sources. By employing advanced machine learning algorithms, the platform generates forecasts based in collective insights, improving overall accuracy.

5.6.5 Market Engine

At the core of the Data Marketplace operations lies the Market Engine. This component is pivotal in orchestrating various functions, including bid validation, data validation, collaborative forecasting, and revenue distribution. It ensures fair trading, validates bids and data, and enhances the reliability of forecasts.

5.6.6 Secure Fund Management through Market Wallet

The platform integrates a public cryptocurrency Market Wallet, enhancing transaction security and transparency between all participants. This wallet validates bids and securely manages cryptocurrency funds, guaranteeing safe and seamless financial transactions.

5.6.7 Future advancements

Anticipating future growth and needs, our development roadmap includes:

- **Scalability and Performance:** Continuous optimization of the platform's scalability and performance as user and data volume increases.
- **Advanced Forecasting Algorithms:** Ongoing research and development will refine forecasting algorithms, incorporating advanced models and domain-specific insights.
- **Enhanced Privacy and Security:** A sustained focus on privacy-preserving techniques ensures data security while promoting collaboration.



- Deeper ENERSHARE Integration: Further integration with ENERSHARE Data Space will enhance data synchronization and collaborative features, offering a holistic ecosystem for renewable energy data management and analysis

5.7 Blockchain

The blockchain is one of the technologies called Distributed Ledger Technology (DLT). It is a peer-to-peer managed chain of blocks that allows information to be stored in a distributed manner among all nodes in the network. The information stored within a blockchain is immutable and protected by cryptography. For these reasons blockchain is very useful for the creation and management of cryptocurrencies. The functionality offered by blockchain reflects the purposes of the international data spaces, in fact, there are some interesting applications of blockchain in the IDS ecosystem¹³.

Among blockchains there are two main groups: permissionless or public blockchains, in which anyone can participate in the creation or validation of new blocks and permissioned or private blockchains, in which access/participation is restricted to a small group of nodes.

Both types of blockchain have pros and cons. While in public blockchains the network is more secure, given the number and diversity of nodes, the transactions inside these blockchains require the use of cryptocurrencies, with a not negligible cost, and the information stored is visible to all. On the other hand, in private blockchains the security and stability given by the presence of many nodes is diminished, while the possibilities of maintaining privacy over the stored information increase. In addition, there is greater freedom in the use of the blockchain itself, with the possibility of defining tokens internal to the blockchain, allowing greater freedom in defining the exchange policies of the Marketplace built on top of it. The blockchains based on Ethereum turn out to be the most versatile in that, using smart contracts, they allow rules, features and controls to be defined that are fully customizable with respect to need.

For these reasons, the blockchain supporting the Marketplace could be developed using a private, Ethereum-based blockchain. These two features, as mentioned before, would allow us to manage a custom token and use smart contracts to customize the functionality offered by the blockchain.

The use of smart contracts in conjunction with a custom token would allow token crediting policies to be defined for Marketplace Participants (e.g., after an asset is published the Participant could receive some tokens) so as to incentivise asset sharing in the Marketplace.

¹³ https://docs.internationaldataspaces.org/ids-ram-4/context-of-the-international-data-spaces/2_1_data-driven-business_ecosystems/2_9_blockchain



A further use of smart contracts could be to generate receipts containing the main information attached to a transaction that took place in the Marketplace. In this way, all transactions and their main information would always be available in the blockchain for periodic or time-to-time checks.



6 Existing implementations

This section will provide information on the existing implementations of architectural components that will be released as ENERSHARE Data Value Stack Alpha version.

6.1 AppStore

6.1.1 Actors

The ENERSHARE App Store design shapes a web-based application that will embody the identified functionalities in Section 5.2. The concept includes 5 key actors, as depicted in Table 30.

Table 30: App Store System Actors

<i>Actors</i>		
<i>Actor Name</i>	<i>Actor Type</i>	<i>Actor Description</i>
App Store	System	App Store Backend System
App Store Frontend	System	App Store Frontend System
IDS Connector	System	IDS compliance connector (integrated as part of Backend)
User	People	The user that interacts with the App Store
Identity Provider	System	Provided and verifier of identity within the data space
Container Registry	System	System that holds App containers available in the App Store.

Each system actor plays a specific role to fulfil a set of scenarios, covering the required actions, what triggers them as to accomplish the functionalities identified and what are the expected results/outputs, being the result of processing data or a trigger to other functionalities.



6.1.2 Scenarios and Diagrams

The following tables summarize the key characteristics for each scenario, namely the event sequence number, the name and description of the event and the related actors. The scenarios covered include:

- The Boot process of the App Store - Table 31.

Publishing a data app

- Table 32.
- Browse and retrieve data apps - Table 33.

Table 31: App Store Boot Process scenario

<i>Scenario name: Boot process</i>					
<i>Step No.</i>	<i>Event</i>	<i>Name of process/ activity</i>	<i>Description of process/ activity</i>	<i>Information producer (actor)</i>	<i>Information receiver (actor)</i>
1.1	App Store Boots	App Store boots	The App store process starts and relays to the container registry and underlying IDS connector to also boot.	App Store	Container Registry, IDS connector
1.2	App Store request identity certificate	Collect Identity Certificate	The App Store requests an identity certificate from the Identity provider.	App Store	Identity Provider
1.3	Identity Provider assigns certificate in the App Store	Provide Identity Certificate	The identity Provider validates the identity request and assigns identity certificate.	Identity provider	App Store





Table 32: App Store Publish Data Apps scenario

<i>Scenario name: Publish Data Apps</i>					
<i>Step No.</i>	<i>Event</i>	<i>Name of process/ activity</i>	<i>Description of process/ activity</i>	<i>Information producer (actor)</i>	<i>Information receiver (actor)</i>
2.1	Send App Container Image	Push App Container to App Store	The IDS connector pushes the app container image to the app store container registry.	IDS Connector	App Store
2.2	Send App Metadata	Send App Metadata to App Store	The IDS Connector sends app metadata to the app store,	IDS Connector	App Store
2.3	Process and store app metadata	Process and store app metadata	The App Store receives and processed app metadata	AppStore	AppStore
2.4	Create app record	Create App record in App Store	An App record is created in the App Store	AppStore	AppStore
2.5	Add app container image to registry	Process and add app container image	The app container image is processed and added to the app store container registry	AppStore	AppStore
2.6	Push app container image	Push container image	The Data App container is pushed to the container registry and is validates.	AppStore	Container Registry





Table 33: App Store Browse and Retrieve Data Apps scenario

<i>Scenario name: Browse and Retrieve Data Apps</i>					
<i>Step No.</i>	<i>Event</i>	<i>Name of process/ activity</i>	<i>Description of process/ activity</i>	<i>Information producer (actor)</i>	<i>Information receiver (actor)</i>
3.1	User searches app	Search Apps	The User navigates the App store UI and searches for the intended Data App	App Store	User
3.2	App Store request App Metadata	Query App list	The App Store frontend reaches for its backend to search for app metadata.	App Store	App Store
3.3	Request for app metadata	Request images Metadata	The App Store contacts its container registry of data apps to request metadata about Data Appa	Container Registry	App Store
3.4	Validate App Metadata	Verify App Containers	The container registry verifies the validity of Data Apps, including those that are certified and those that are not certified.	Container Registry	Container Registry
3.5	Data App is selected for download	Select Data App	A specific Data App is marked for download.	App Store	User
3.6	Data App is selected for download	Select Data App	A specific Data App is marked for download.	App Store	App Store
3.7	Data App is selected for	Request Data App	A given data app container is found and selected for	Registry	App Store



	download		download		
3.8	Data App is set for validation	Validate Data App	Before provisioning, a Data App is validated.	Registry	Registry
3.9	Data App is set for collection	Collect Data App	Data Container is retrieved from registry	Registry	Registry

The detailed scenarios shape the overall diagram of all interactions that make up the App Store system. These are depicted in Figure 7.

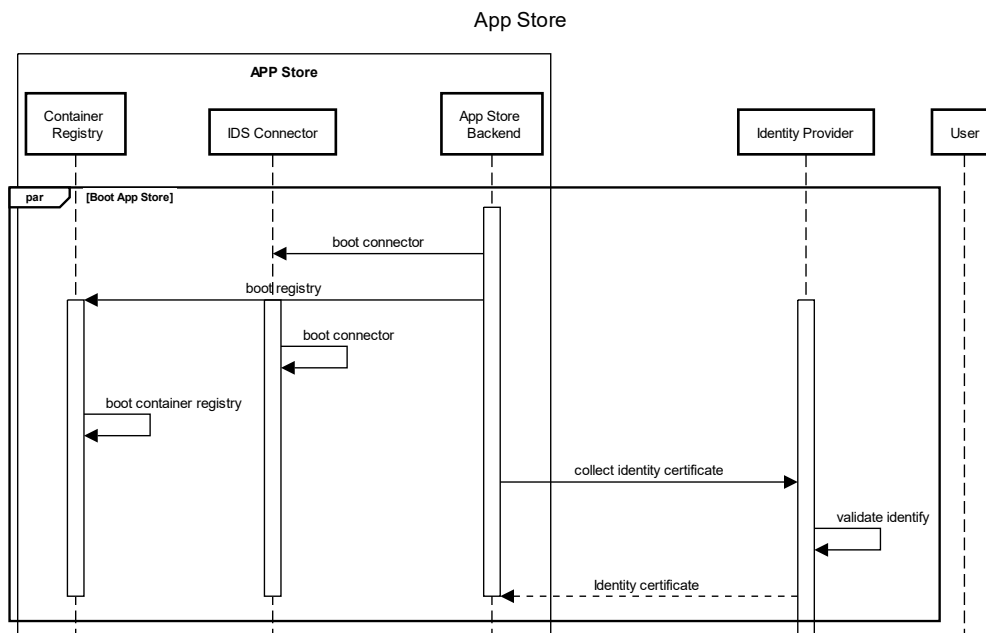


Figure 7: App Store boot process sequence diagram



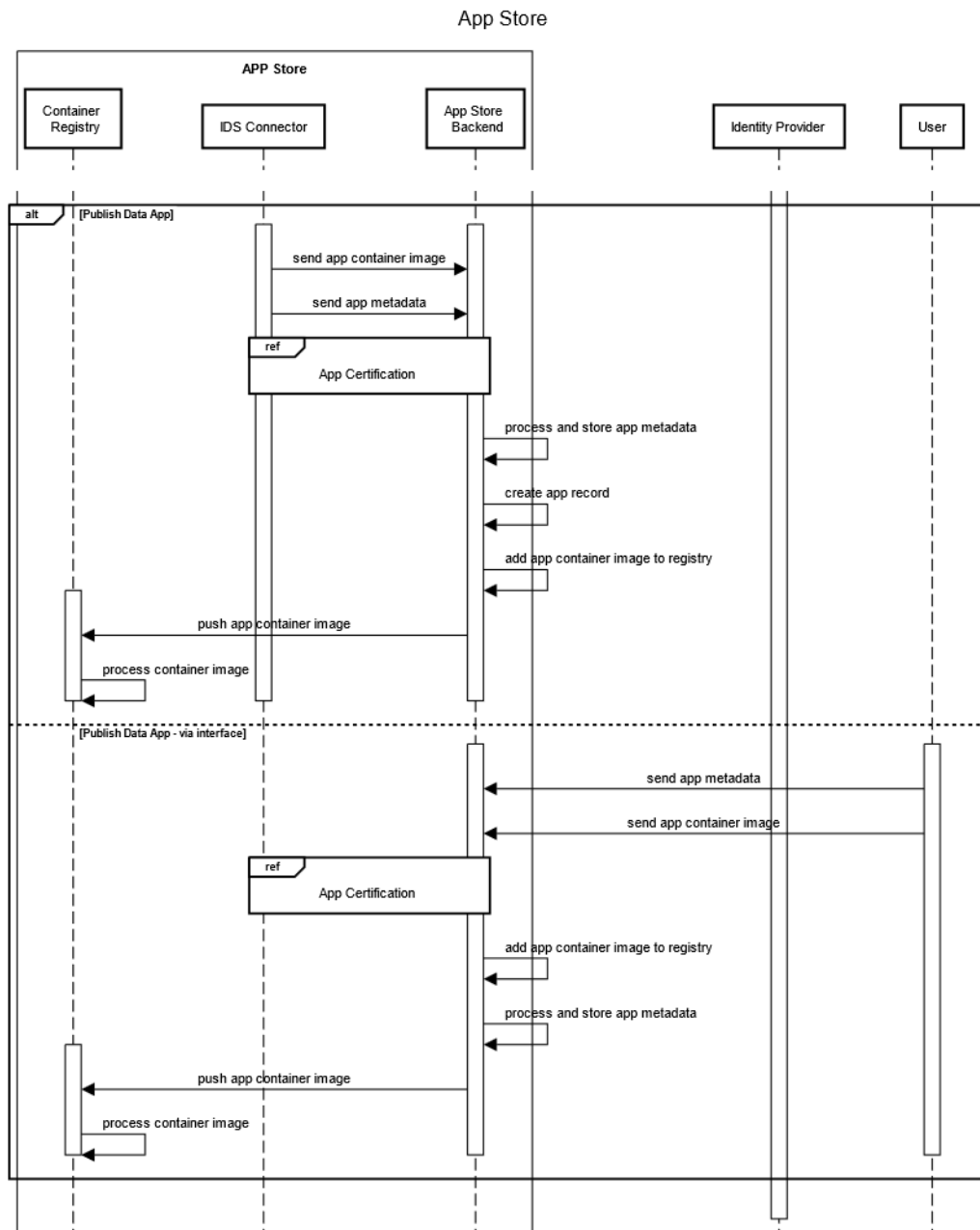


Figure 8: App store publish data process sequence diagram

6.1.3 App Store Architecture

The architecture of the App Store is depicted in Figure 9. It is split in two sub-components, the App Store Backend and the App Store Frontend. As a web-based application, the App Store architecture follows a Model-View-Controller design pattern where the Model and Controller



dimensions are in scope of the backend sub-component, while the View dimensions is in scope of the frontend sub-component.

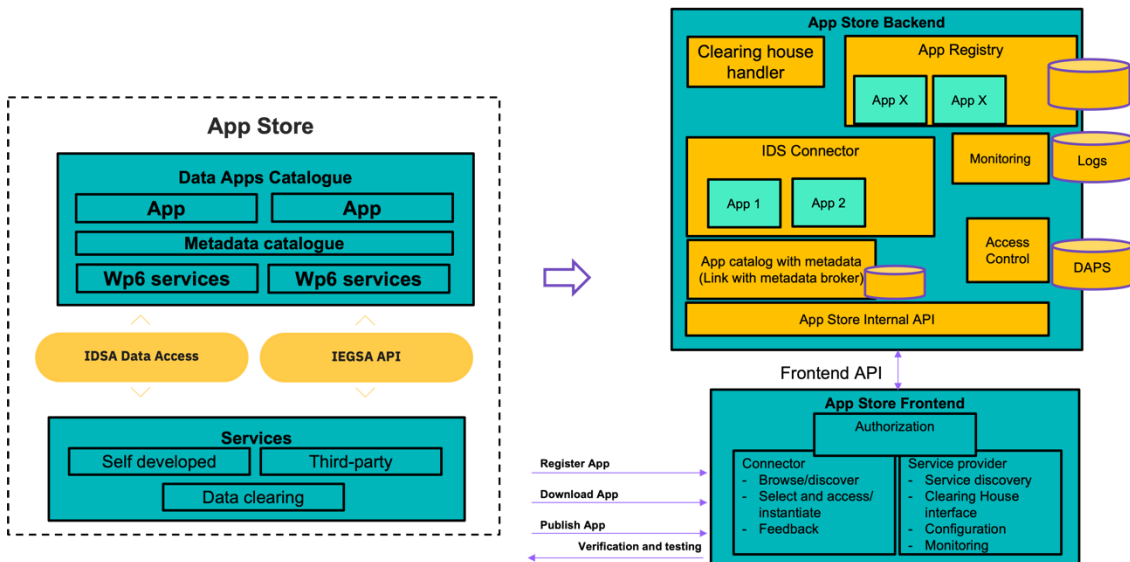


Figure 9: App Store Architecture

The App Store backend departs from an IDS compliant connector, providing the key integration with a data space instantiation. The embedded connector will link with identity provisioning of the App Store itself, also taking part in the identity validation for all other interactions with foreign data connectors.

As a catalogue of available Apps, the App Store includes an App registry module, holding the container images of apps as assembled by their owners. The registry only links with the App Store backend system that forwards it to the data connector interface. Handling metadata that stems from the App descriptions is currently handled by the App Store internal data model as a support for the application development. The next release will move it forward and consider the use of the metadata broker as means to support management and querying of data assets. The next release will also consider the Clearing House as a complement to assist the process where a payment is considered to unlock the transfer process of a data app. Finally, other modules assist the workflow of using the App Store, as the monitoring module that records operations or the access control module that links with the DAPS system of the data space.

The frontend sub-component provides user interfaces to browse the catalogue of apps, upload and download apps. Moreover, it also assists the process of uploading app metadata (key for searching apps).



6.2 Clearing House

The ENERSHARE Clearing House (CH) considers the existing implementations namely the Fraunhofer-AISEC Clearing House¹⁴ and the OneNet Connector¹⁵. The latest version of the IDS RAM 4.0 prescribes that the Clearing House shall base all its functions on a logging service that records information relevant for clearing and billing as well as usage control¹. The complete set of features of a Clearing House shall encompass the Clearing and Settlement Service, Logging Service, Billing Service and UC Claim Validation Service, as illustrated in Figure 10.

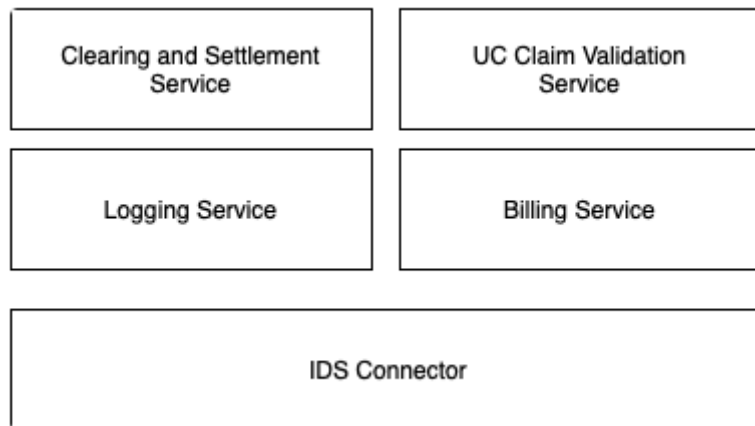


Figure 10: Clearing House Architecture

Fraunhofer-AISEC Clearing House

Current implementation following the IDS RAM principles are openly provided by Fraunhofer-AISEC Clearing House, which solely provides logging features. The core idea relies on storing transaction logs in encrypted and immutable manner. The data immutability is achieved by storing in the Clearing House databased log entries including a hash value of the preceding log entry, hence, chaining together all log entries. Any change to a previous log entry would require rehashing all following log entries. The Connector logging information in the Clearing House receives a signed receipt from the Clearing House that includes among other things a timestamp

¹⁴ <https://github.com/Fraunhofer-AISEC/ids-clearing-house-service/tree/master/clearing-house-app>

¹⁵ <https://github.com/european-dynamics-rnd/OneNet>



and the current chain hash. A single valid receipt in possession of any Connector is enough to detect any change to data up to the time indicated in the receipt.

OneNet implementation

The OneNet project has developed the OneNet Connector by extending the core developments of the FIWARE True Connector¹⁶. The OneNet Connector aims to enable a European Energy Data Space, combining the IDS principles with the advantages of the FIWARE ecosystem ensuring a seamless and secure data exchange in a completely end-to-end decentralized approach. The OneNet Connector is ready to be deployed and integrated in any existing platform and offers user-friendly interfaces (both as REST APIs and GUI) enabling users and platforms to share data.

At the Clearing House level, integration with the Fraunhofer Clearing House has been achieved covering the logging services for the executed data exchanges. On top of the IDS layer, a local GUI application accommodates custom developments that are realizing project, business and domain specific functionalities (i.e., referring to establishing the Cross-Platform services which are applying a service-based scheme under which data transactions are allowed solely upon active service subscription). The local application stores various meta-data (e.g., service contracts, user preferences) including logging of data transactions with other peer connectors. All such information is communicated to the OneNet Decentralized Middleware which concurrently maintains relevant log entries referring to the data transactions. An illustration of stored OneNet's Data exchange timeline covering primary meta-data information on the logs is presented on Figure 11.

16 https://fiware-true-connector.readthedocs.io/en/latest/true_connector_tutorial.html



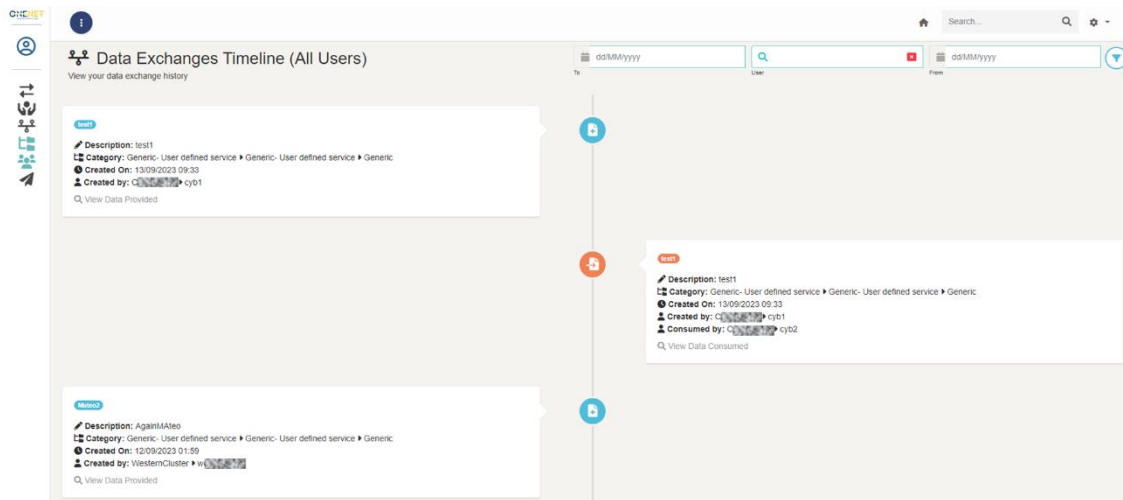


Figure 11: OneNet data exchanges Timeline providing primary meta-data logging.

6.3 Metadata Broker

The TNO Security Gateway (TSG) Metadata Broker¹⁷ is considered as a candidate for the ENERSHARE Metadata Broker. The TSG Metadata Broker is an IDS connector that contains endpoints for the registration, publication, maintenance, and query of self-descriptions. The TSG Metadata Broker is one of several TSG components that allow a party to participate in an IDS data space.

6.3.1 TSG Components

The TSG components are based on the IDS reference architecture model version 4 (IDS-RAM 4)¹⁸ and implement its main features, namely:

Trust, the basis of the International Data Spaces, by means of evaluation and certification before getting granted access to the dataspace.

Security & Data Sovereignty, for Secure data exchange with the certainty of remaining in control of your data, by means of security protocols and policy enforcement.

Ecosystem of Data, where decentralization is a key aspect of International Data Spaces, keeping the data as close to the source as possible and only sharing it when explicitly allowed.

Standardized interoperability that uses standardized communication patterns between IDS Connectors, which allow different connector implementations.

¹⁷ <https://tno-tsg.gitlab.io/docs/overview/>

¹⁸ <https://docs.internationaldataspaces.org/ids-ram-4/>



Value Adding Apps to create a modular ecosystem in which value adding applications can be added on-demand to IDS connectors.

Data Markets, where data-driven services combined with the pillars above allow for new business models to thrive.

In addition, the TSG components have the following features:

Kubernetes support: All of the TSG components are created with support for Kubernetes in mind, that make the deployments of components robust and allow for production-ready deployments.

Abstractions of IDS specifics: By means of specific Data Apps developed to enable the inclusion of existing systems.

Embedded Policy Enforcement: The TSG Core Container contains an embedded Policy Enforcement Framework that supports a variety of IDS Usage Policies.

Easy Configuration: The TSG components can be deployed in a wide variety of use cases, configuration of the components receives special care to prevent overwhelming configuration and allow the flexibility of the configuration.

The TSG Core Container implements the main features of the IDS-RAM. Next to the Core Container are Data Apps that extend the Core Container and implement business logic. The Core Container and Data App together form a TSG Connector.

6.3.2 TSG Metadata Broker

The TSG Metadata Broker is implemented as a TSG Connector (see 6.3.1). The key function of the Metadata Broker is to enable publication and discovery of services by providing the metadata of connectors within the dataspace.

Providers of services publish metadata about their services in the TSG Metadata Broker. Once published, consumers can query the TSG Metadata Broker to find best-matching services. The metadata also contains the information required for the consumer to connect to the provider and initiate the data transaction. In other words, the TSG Metadata Broker provides the consumer with the necessary information to discover and select a service and to set up a data transaction using that service. The TSG Metadata Broker does not play a role in the actual data transaction.

The connectors in the dataspace are responsible for making sure the self-descriptions in the TSG Metadata Broker are up to date. The self-descriptions can be accessed by connectors via querying (e.g., via SPARQL). The TSG Metadata Broker uses a triple store as its backend to



enable the SPARQL query functionality. The IDS Information Model is used for the messages format.

The TSG Metadata Broker is currently at technology readiness level (TRL) 7.

6.3.3 Message flows

The message flows of the TSG Metadata Broker implement the IDS standard, using the IDS Information Model and message patterns.

6.3.3.1 Publish Self-Description

A connector publishes a self-description to the TSG Metadata Broker so that it becomes discoverable for other connectors in the data space. The publication request sent to the TSG Metadata Broker consists of a *ConnectorUpdateMessage* with a self-description as payload. The trigger for the Core Container to perform a publication request is either a fixed time interval or a change event of the metadata (Figure 12).



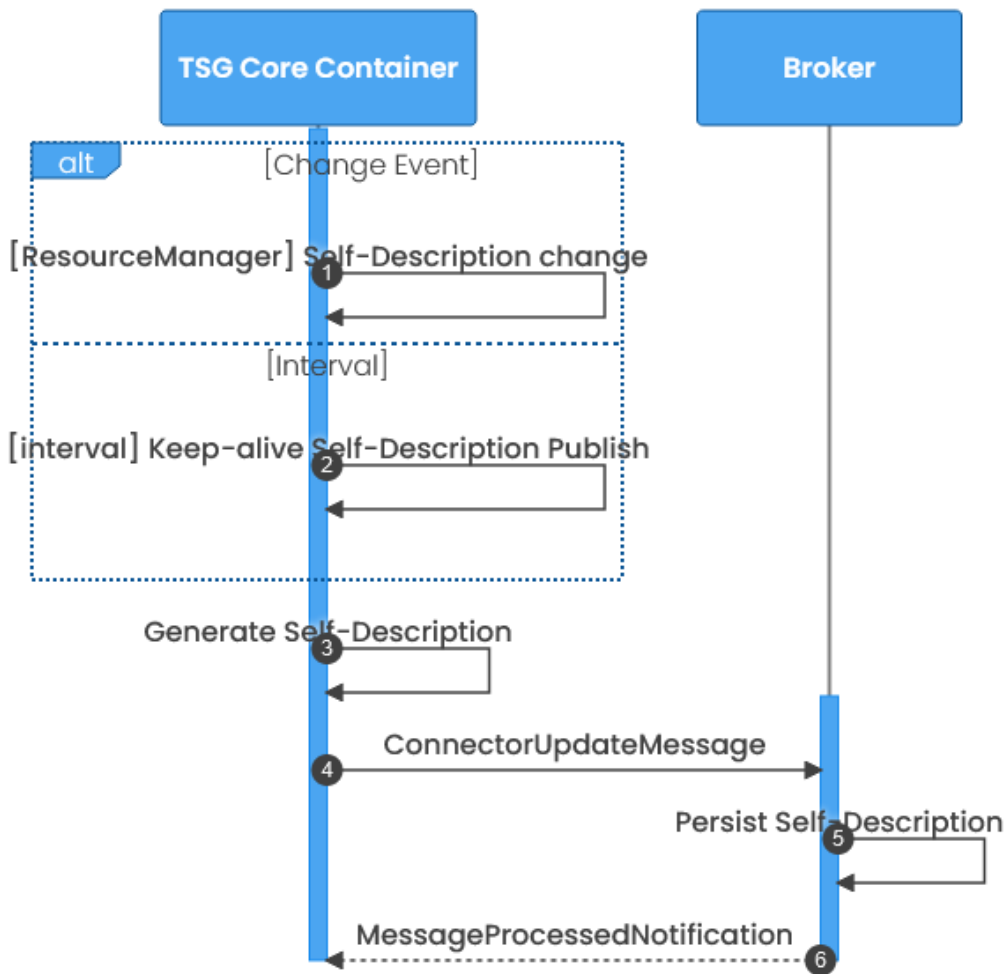


Figure 12: Publish self-description message flow

6.3.3.2 Query self-descriptions

Consumers can query the self-descriptions in the Metadata Broker. The message sent to the Broker is of type *QueryMessage* with a SPARQL query as its payload. The response is in JSON-LD format that can be parsed by the TSG components (Figure 13).



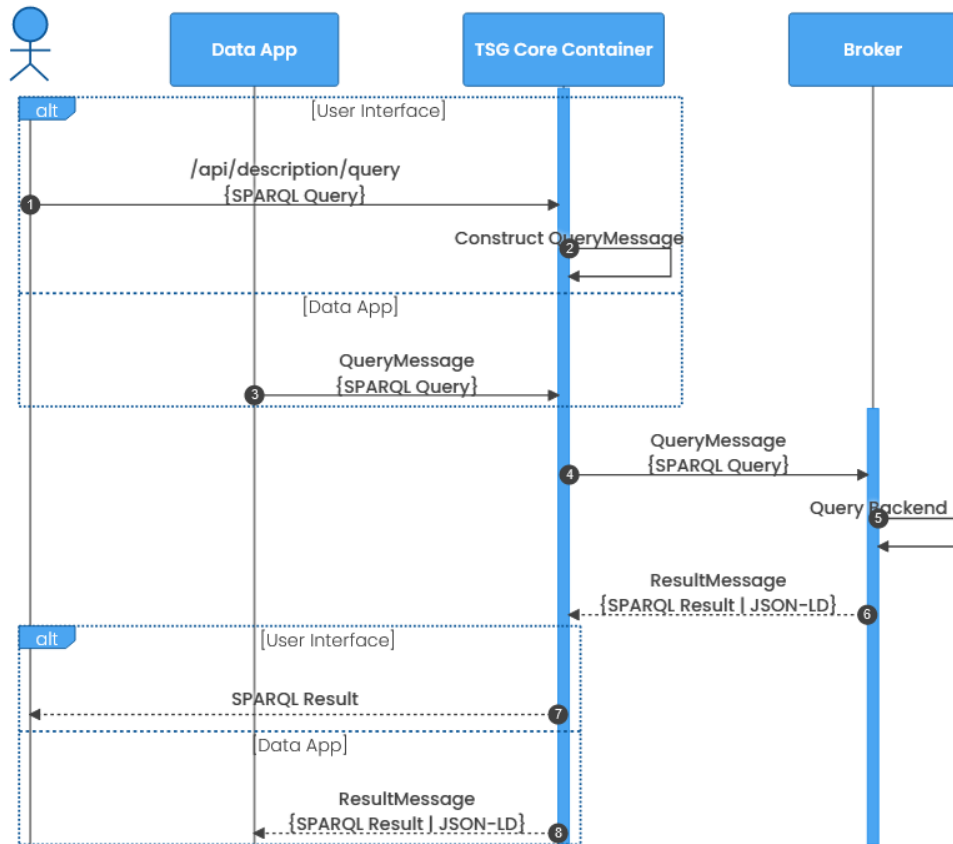


Figure 13: Query self-description message flow

6.4 Data Monetization and Barter Sharing Incentive Module

6.4.1 Input and Output Data Format

The data sharing incentive module (or data Marketplace) follows a standardized data format and provides an API for data transfer and integration. The I/O data format and APIs are designed to ensure compatibility and easy integration with various systems and applications.

6.4.2 Data Format

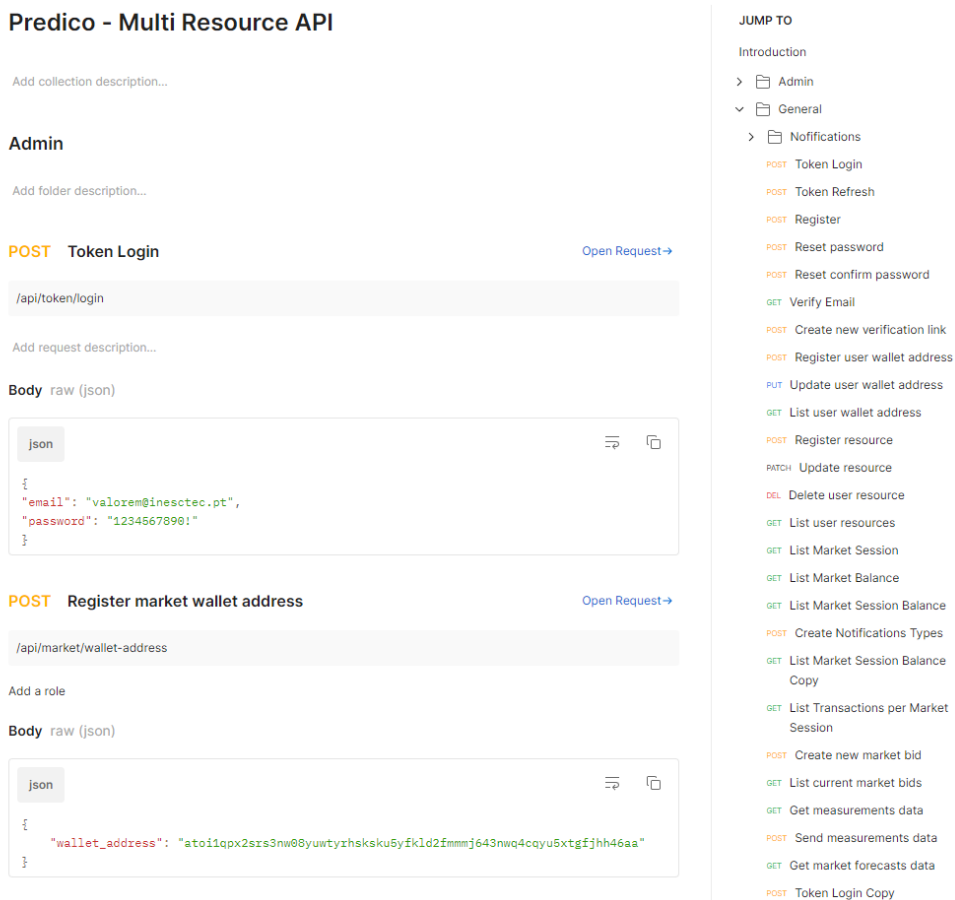
The data format used in the data Marketplace is based on industry-standard formats such as JSON (JavaScript Object Notation) and CSV (Comma-Separated Values). These formats allow for easy representation and exchange of structured data for forecasts and energy production datasets.



For data contributions, agents can upload their data in CSV format from the Application UI management section where each column represents a specific data attribute such as resource identification, timestamp, and power production. JSON format is the most widely adopted and can be used for the service API to transmit data in a structured manner, allowing efficient download and upload of data.

6.4.3 API – Requests and Structures

As mentioned throughout the course of this report, the Data Marketplace provides a comprehensive API (Figure 14) to facilitate data transfer, querying, and collaborative forecasting.



Predico - Multi Resource API

Add collection description...

Admin

Add folder description...

POST Token Login [Open Request →](#)

/api/token/login

Add request description...

Body raw (json)

```
json
{
  "email": "valorem@inesctec.pt",
  "password": "1234567890!"
}
```

POST Register market wallet address [Open Request →](#)

/api/market/wallet-address

Add a role

Body raw (json)

```
json
{
  "wallet_address": "atoi1qpx2szs3nw08yurtyzhsksku5yfkld2fmmjmj643nwq4cqyu5xtgfjhh46aa"
}
```

JUMP TO

- Introduction
- > Admin
- > General
- > Notifications
 - POST Token Login
 - POST Token Refresh
 - POST Register
 - POST Reset password
 - POST Reset confirm password
 - GET Verify Email
 - POST Create new verification link
 - POST Register user wallet address
 - PUT Update user wallet address
 - GET List user wallet address
 - POST Register resource
 - PATCH Update resource
 - DEL Delete user resource
 - GET List user resources
 - GET List Market Session
 - GET List Market Balance
 - GET List Market Session Balance
 - POST Create Notifications Types
 - GET List Market Session Balance Copy
 - GET List Transactions per Market Session
 - POST Create new market bid
 - GET List current market bids
 - GET Get measurements data
 - POST Send measurements data
 - GET Get market forecasts data
 - POST Token Login Copy

Figure 14: API documentation overview

The API documentation overview (Figure 14) provides a comprehensive view of the Data Marketplace API, showcasing the available endpoints, parameters, and response formats. This valuable resource offers detailed technical information on how developers can effectively utilize the API to automate essential functionalities such as bids, data uploads, and forecast downloads.



The online documentation can be found in the following address:

<https://documenter.getpostman.com/view/391645/2s93z9a2Ui>

Some of the main features offered by the API include:

1. **Data Upload:** The API allows data providers to securely upload their renewable energy production datasets to the platform. The API supports batch uploads, enabling users to transfer large volumes of data efficiently.
2. **Forecast Query:** Users can query and retrieve specific datasets or subsets of forecast data using the API. This allows users to specify criteria such as resource and time range attributes to retrieve relevant information for their analysis and forecasting needs.
3. **Session Bid:** The API allows users to automate their bids and transactions within the data Marketplace. Users can use a dedicated library provided by the Marketplace, in association a library to securely post their bid transactions to the Marketplace. This automation streamlines the bidding process, making it more efficient and enabling users to maximize the value of their data contributions, forecasts, and rewards.
4. **User Management API:** The User Management API provides functionality for user authentication, registration, and account management. It allows users to create and manage their profiles, access tokens, and configure data contribution settings.

6.4.3.1 Session Bid Request

The following snippet illustrates the “Session Bid” query and how more advanced agents (developers) can develop a script using custom libraries provided by the Data Marketplace to issue a bid and therefore automate the process of bidding in market sessions:

Table 34: API integration bid script example

```
Session Bidding
from predico_Marketplace import DataMarketplace
from iota import Iota, ProposedTransaction, Address

# Initialize the Data Marketplace with the required Token that can be extracted in the Application UI
# settings and retrieve the current open session. Note: other relevant metadata information about the
# session could be obtained through the get_session() method

data_Marketplace = DataMarketplace(api_token="YOUR_API_TOKEN")
session_id = data_Marketplace.get_session(status="OPEN")["session_id"]

# Connect to the IOTA network – It’s required to use IOTA library for this step
api = Iota("https://nodes.devnet.iota.org:443")

# Generate a new IOTA address for the user's wallet
```





```
seed = "YOUR_IOTA_SEED"
address = Address(
    'PREDICO_MARKET_WALLET_ADDRESS',
    key_index=0,
    security_level=2
)

# Create a new transaction with the value of the bid
bid = 100
transaction = ProposedTransaction(
    address=address,
    value=bid, # Specify the value of IOTA tokens for the bid
    message=data_Marketplace.generate_bid_message("BID_ID", "SELLER_ID")
)

# Sign and send the transaction
signed_transaction = api.prepare_transfer(seed, [transaction])
api.send_trytes(signed_transaction["trytes"])

# Wait for the transaction to be confirmed
confirmed = False
while not confirmed:
    bundle = api.find_transactions(addresses=[address])
    confirmed = len(bundle["hashes"]) > 0

# Transaction confirmed, get the transaction hash
transaction_hash = bundle["hashes"][0]

# Post the transaction hash to the data Marketplace session bid
# by default the call is issued to the current open session.
response = data_Marketplace.session_bid(session_id=session_id, bid_value=bid, transaction_id=
transaction_hash)

# Print the response from the Data Marketplace
print(response.json())
```

The provided code demonstrates the process of making a bid on the Data Marketplace using the Data Marketplace Python library and the IOTA library. The following outlines the steps:

1. Initialization: The code initializes the Data Marketplace by creating an instance of the DataMarketplace class and providing the required API token. This token is obtained from the Application UI settings.
2. Connection to the IOTA Network: The code establishes a connection to the IOTA network using the IOTA wallet binding for python. It specifies the network URL, in this case, "https://nodes.devnet.iota.org:443".



3. **Generating an IOTA Address:** A new IOTA address is generated for the user's wallet. This address is used to send the bid transaction. In this case it should be the Market Wallet address.
4. **Creating the Bid Transaction:** The code creates a new transaction for the bid. It specifies the recipient address (the Data Marketplace's wallet address), the value of the bid (in IOTA tokens), and includes a message generated by the Data Marketplace library to identify the bid with the given bid ID and buyer ID.
5. **Signing and Sending the Transaction:** The bid transaction is signed with the user's IOTA seed and sent to the IOTA network using the `prepare_transfer` and `send_trytes` methods of the IOTA library.
6. **Waiting for Confirmation:** The code waits for the bid transaction to be confirmed on the IOTA network. It continuously checks the status of the transaction by finding transactions associated with the generated address until confirmation is received.
7. **Retrieving the Transaction Hash:** Once the bid transaction is confirmed, the transaction hash is obtained from the bundle of transactions associated with the generated address.
8. **Posting the Transaction Hash to the Data Marketplace:** The transaction hash is then posted to the data Marketplace's session bid using the `session_bid()` method of the `DataMarketplace` instance. The Data Marketplace Backoffice will then validate the bid for the current open session and return a response.
9. **Retrieving and Printing the Response:** The response from the Data Marketplace is retrieved and printed, providing information about the bid and its status.

Overall, the code shows the process of making a bid on the Data Marketplace by generating an IOTA transaction, waiting for its confirmation, and posting the transaction identifier to the data Marketplace API. This allows users to participate in the bidding process of the Data Marketplace and interact with the platform programmatically.

Note: The Desktop Application uses the same underline logic, using the cryptocurrency library and API endpoints for Market bid interactions on the user's behalf. For security reasons a PIN code is necessary to issue a transaction and therefore automated bids are not possible in the frontend application since storing the PIN code in the application memory may result in a security risk for the user.

To assist developers in utilizing the bid functionality, comprehensive guides and documentation will be available in the FAQ section of the Desktop Application along with other web resources. These resources provide detailed explanations of the steps involved in making a bid, including the required authentication, transaction generation, monitoring, and interaction with the Data Marketplace API. The guide aims to facilitate the integration of the bid functionality into custom applications and enable developers to leverage the full potential of the Data Marketplace platform.



Note: The Data Marketplace platform provides a sandbox environment where developers can familiarize themselves with the bidding process and interact with the platform without engaging in real transactions. The sandbox environment allows users to experiment, test their code, and gain a better understanding of the platform's functionalities before engaging in live bidding.

6.4.3.2 Data Management Request

The REST API supports both JSON, XML, CSV formats for communication. Agents can use the API to automate their data uploads in a structured format. The following table provides an example of the fields and endpoints requests for contributing with data and download of forecasts respectively:

Table 35: API requests examples for uploading and downloading of data

/api/data/raw-data [POST]	api/data/market-forecasts?resource_name=resource-1 [GET]
<pre>{ "user": "352de502-a118", "resource_name": "string", "time_interval": "5", "aggregation_type": "avg", "units": "w", "timeseries": [{ "datetime": "2023-04-04T15:53:53.342Z", "value": 0 }], }</pre>	<pre>{ "data": [{ "datetime": "2022-10-01T00:34:58Z", "value": 7802, "units": "kw", "resource": 1, "resource_type": "measurements", "time_interval": "60", "aggregation_type": "avg", "registered_at": "2022-10-20T11:34:58.131623Z" }] }</pre>

The data transfer architecture of the Data Marketplace is designed to ensure secure, reliable, and efficient data exchange between data providers and consumers. The architecture includes the following components:

1. **Secure Data Transfer Protocols:** Data uploads and downloads are performed using secure transfer protocols such as HTTPS (HTTP over SSL/TLS) to encrypt data during transmission. This ensures the confidentiality and integrity of the data during transit.
2. **Data Encryption:** To enhance data security, sensitive information is encrypted before storage. This ensures that even if unauthorized access occurs, the data remains protected and unreadable.
3. **Batch Processing:** The data transfer architecture includes batch processing capabilities, allowing users to upload and download data in large volumes. This facilitates efficient



data transfers, especially when dealing with extensive datasets from multiple renewable energy sources.

4. Asynchronous Processing: To optimize performance, the data transfer architecture leverages asynchronous processing techniques. This allows concurrent handling of multiple data transfer requests, enhancing system responsiveness and scalability.

6.4.4 Running the Service

6.4.4.1 Desktop Application

From a regular user perspective, getting started with the data Marketplace service involves signing up for an account on the Desktop Application. Once registered, agents can access the use the user-friendly Desktop Application to configure their data contribution and setting up their bidding preferences.

6.4.4.2 Developers

On the developer side, the process begins with creating an account and obtaining the API token directly from the API. In addition, it's required an IOTA wallet that must be created using the official IOTA library (<https://github.com/iotaedger/wallet.rs>).

Developers can leverage the API token to authenticate their requests to the data Marketplace API and perform a wide range of operations programmatically. By referring to the comprehensive API documentation provided (Figure 14), developers can obtain valuable insights into the available endpoints, request structures, and response formats. This empowers them to integrate the data Marketplace functionalities into their own application systems, enabling custom data analysis workflows, data retrieval, collaborative forecasting, and more.

6.4.4.3 System administrator (Market Operator)

For system administrators responsible for running the data Marketplace platform, deployment is simplified using Docker containers. The platform can be set up following a series of steps to ensure smooth operation and scalability. These steps encompass containerizing the necessary components, such as the backend infrastructure, database, and any additional services. By using Docker, system administrators can streamline the deployment process and ensure consistency across different environments, facilitating efficient management and maintenance of the data Marketplace platform.

The following steps outline the process:

1. Docker Setup: Docker must be installed and properly configured on the deployment environment. Docker provides containerization technology, allowing for consistent and portable deployment across different environments.



2. **Containerization:** Each component of the data Marketplace service should be containerized using Docker. This involves creating Docker containers for the Django application, PostgreSQL database, and Cassandra database. Docker containers encapsulate the necessary dependencies and configurations, providing a standardized and isolated environment for each component.
3. **Docker Compose:** Docker Compose, a tool for defining and managing multi-container Docker applications, can be used to orchestrate the deployment of the data Marketplace service. A Docker Compose file must be created, specifying the configuration of the service's containers, networks, and volumes.
4. **Service Configuration:** The Docker Compose file should be configured to include the necessary services and their dependencies as previously stated. Services should be defined for the Django application, PostgreSQL database, Cassandra database, and any other required components such as load balancers or reverse proxies that may be added.
5. **Networking:** The networking within the Docker Compose file should be configured to allow communication between the different containers. Network aliases or bridge networks can be specified to enable seamless connectivity between the Django application and the databases.
6. **Environment Variables:** Environment variables should be set up within the Docker Compose file or using an environment file to provide necessary configuration parameters such as database credentials, API keys, and other service-specific settings.
7. **Deployment:** The data Marketplace service can be deployed using the Docker Compose command. This command creates and starts the containers based on the configuration specified in the Docker Compose file. Docker Compose handles the initialization and networking of the containers, ensuring they are interconnected and operational.
8. **Monitoring and Logging:** Monitoring and logging solutions should be implemented to track the performance and health of the deployed containers. Tools such as Docker logs, container monitoring systems, or application performance monitoring (APM) platforms can be utilized to gather insights and diagnose any issues.
9. **Continuous Integration and Deployment (CI/CD):** Not required but highly advised setting up a CI/CD pipeline to automate the build, testing, and deployment of the data Marketplace service using Docker. This ensures a consistent and reliable release cycle, streamlining the development and deployment process.

6.4.5 Technology Readiness Level (TRL)

The current Technology Readiness Level (TRL) of the Data Marketplace is at a mature development stage, around TRL 5. The platform is undergoing an extensive development, testing and refinement to ensure its functionality, reliability, and usability. The core features and functionalities have been implemented and validated through multiple iterations and in-



house case studies. The platform is currently in an automated testing environment with a set of mocking agents - including energy providers and forecast buyers.

For the future, it is anticipated that the platform will reach TRL 6 in ENERSHARE deliverables D6.2 and D6.3. This will involve further scalability improvements, performance optimizations and integration with additional data sources, namely the Integration with ENERSHARE Data Space (6.4.7).

We aim to expand the user base and establish partnerships with more renewable energy data providers, thereby enhancing the Marketplace's data offerings and collaborative forecasting capabilities. Continuous monitoring, evaluation, and feedback will be carried out to further improve the platform and address any potential limitations or challenges.

It's important to note that the current Data Marketplace operates in a traditional server-client architecture, with separate server and client APIs (Figure 15). However, our future development is focus on integrating the system with the ENERSHARE Dataspace, which will bring additional capabilities and improve the overall functionality and security of the platform.

6.4.6 Software Details

The data Marketplace is built on a robust and scalable software stack that uses a combination of third-party software and custom-built components. The key software details include:

1. **Programming Language:** The platform is primarily developed using Python, a versatile and widely adopted programming language for data analysis, machine learning, and web development.
2. **Frontend Framework:** The frontend application is developed using JavaScript and the React framework. JavaScript provides the foundation for dynamic and interactive web interfaces, while React offers a component-based approach for building reusable and responsive user interfaces.
3. **Electron:** The data Marketplace frontend also uses Electron, a framework for building cross-platform desktop applications using web technologies. Electron enables the development of desktop client applications using HTML, CSS, and JavaScript, allowing for seamless integration with the data Marketplace's functionalities.
4. **Backend Framework:** Django, a powerful web framework for Python, is utilized to develop the application's backend infrastructure and APIs. Django provides a comprehensive set of tools and features for building robust and scalable web applications. It offers a high-level of abstraction, facilitating rapid development and seamless integration with various components of the application.
5. **Database:** The platform employs a combination of databases to handle different types of data. PostgreSQL is used as a reliable and robust relational database management system for storing and managing various data, including user information, resources



and transaction records. Additionally, Cassandra, a highly scalable and distributed database, is used to handle large volumes of data coming from energy providers. Cassandra's distributed architecture allows efficient storage and retrieval of large volumes of data, making it suitable for handling high-volume data generated by renewable energy sources.

6. Docker: The platform is containerized using Docker, which allows for easy deployment, scalability, and portability across different environments. Docker enables efficient management of dependencies and simplifies the deployment process for the platform.
7. Job Scheduling: To automate various data-related tasks and ensure timely updates, the platform uses job scheduling tools like Celery. This tool facilitates the scheduling and execution of periodic collaborative forecasting processes, and other background tasks such as user and email and logging notifications.
8. Licensing: The third-party software components used in the platform adhere to their respective licenses, including open-source licenses such as MIT, Apache, and GNU GPL. We ensure compliance with license requirements and promote the use of open-source software within our platform.

6.4.7 Integration with ENERSHARE Data Space

The integration of the Data Marketplace with the ENERSHARE Dataspace can bring additional value to the data being traded. The Dataspace serves as a decentralized infrastructure that enables secure and controlled access to data from various sources. When combined with the Data Marketplace, it offers enhanced features for data valuation. Here's how the integration with the Dataspace can contribute to the valuation of data:

1. Data Provenance and Trust: The Dataspace provides mechanisms for tracking and verifying the provenance of data. This includes information about its origin, ownership, and the processes it has undergone. By integrating with the Dataspace, the Data Marketplace can leverage this provenance information to assess the trustworthiness and reliability of data sources. Data with transparent and verifiable provenance tends to have higher value due to increased trust and accountability.
2. Access Control and Security: The Dataspace enables fine-grained access control and data security measures. When data in the Data Marketplace is integrated with the Dataspace, it can benefit from advanced access control mechanisms, such as attribute-based access control (ABAC) or policy-based access control (PBAC). These mechanisms allow data providers to define access policies and permissions, ensuring that only authorized and authenticated users can access the data. The enhanced security and controlled access to data contribute to its value.
3. Enhanced Metadata and Contextual Information: The Dataspace allows for the enrichment of data with metadata and contextual information. This metadata includes descriptions, annotations, and semantic tags that provide additional insights and



context to the data. By integrating with the Dataspace, the Data Marketplace can leverage this enriched metadata to provide more detailed and accurate information about the data being traded. Enhanced metadata contributes to better data discovery, understanding, and ultimately, its value.

In sum, the integration with the Dataspace brings added benefits to the Data Marketplace by enhancing data provenance, access control, security, governance, collaboration, and metadata. These factors collectively contribute to the valuation of data, enabling more informed decision-making, fostering trust among stakeholders, and promoting the exchange of high-quality and valuable data within the Marketplace.

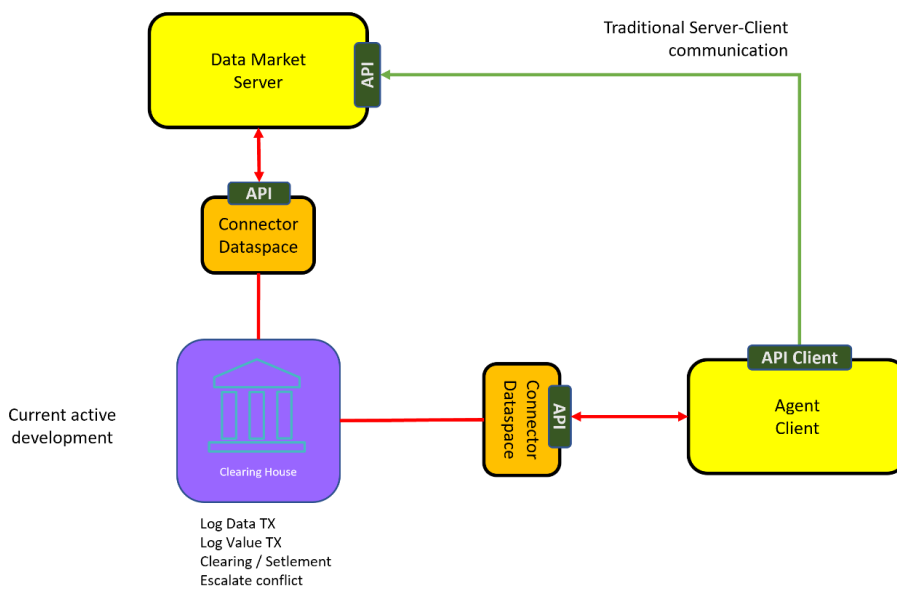


Figure 15: Traditional server-client communication vs decentralized communication

Figure 15 illustrates the fundamental difference between traditional server-client communication and a decentralized secure data exchange in the context of a Data Marketplace.

In the traditional server-client communication model, a central server acts as the intermediary between data providers and consumers. The server holds the data and facilitates data transactions with clients (consumers) directly connecting to the server to request and access the data. This centralized approach relies on a single point of control and requires trust in the server to handle data securely and manage transactions reliably.

In contrast, the decentralized data exchange model introduces a more robust and transparent framework for data exchange. The Clearing House and Connectors attached play a key role in this decentralized approach. The Clearing House acts as a trusted intermediary and maintains a transparent record of data and value transactions. It ensures integrity, accountability, and fairness in the Data Market ecosystem. The Connectors, integrated with the Clearing House,



enable secure and seamless data exchange between data providers and consumers while preserving privacy and confidentiality.

In this decentralized model, Data Providers and Data Consumers interact with the Data Marketplace and with each other through the Connectors. The Connectors facilitate data uploads, retrievals, and transactions while leveraging the services of the Clearing House for validation and verification. The Clearing House acts as a transparent and trusted entity, ensuring that data transactions are logged, value transactions are recorded, clearing and settlement occur accurately, and conflicts are resolved efficiently.

The decentralized data exchange model enhances trust, privacy, and transparency in the Data Market. It distributes control and authority across the ecosystem, reducing reliance on a central server and promoting a more secure and collaborative data exchange environment.

The following are the expected features of the Clearing House:

1. **Log Data Transactions:** The Clearing House logs all data transactions that occur within the Data Market. This includes the details of data uploads, retrievals, and any modifications made to the data. By maintaining a comprehensive record of these transactions, the Clearing House enables traceability and transparency, ensuring that data providers and consumers have a clear audit trail of their interactions.
2. **Log Value Transactions:** In addition to data transactions, the Clearing House also logs value transactions related to payments and bids. It records the financial aspects of the Data Marketplace, including the payment transactions made to data providers and the bids made by consumers to acquire forecast datasets and other data. By logging value transactions, the Clearing House ensures transparency in financial activities within the Data Marketplace.
3. **Clearing / Settlement:** The Clearing House facilitates the clearing and settlement of transactions between data providers and consumers. It ensures that the agreed-upon payments are processed securely and accurately. The Clearing House verifies the completion of data transactions and confirms the settlement of financial obligations between parties involved. This process helps to maintain trust and efficiency in the Data Market by ensuring timely and accurate financial settlements.
4. **Escalate Conflict:** In the event of a dispute or conflict arising between data providers and consumers, the Clearing House serves as an arbiter and mediator. It provides a mechanism to escalate and resolve conflicts by facilitating communication and negotiation between the involved parties. The Clearing House ensures that any conflicts or discrepancies are addressed promptly and fairly, fostering a trustworthy and reliable environment for data exchange.



The IDS Testbed under development provides a foundation for this integration, offering various features such as the True Connector, TSG Metadata broker, TSG Connector, IDS Connector, Certificate Authority, and DAPS. These components work together to ensure secure and trusted data exchange between the Data Marketplace and the Dataspace.

The True Connector serves as a bridge between the Data Marketplace, consumers providers and the Dataspace, allowing communication and data transfer between all parties. It can facilitate the interaction with the TSG Metadata broker, which provides metadata management capabilities to enable efficient publication and discovery of data resources. Similarly, the TSG Connector and IDS Connector can execute similar roles in establishing secure connections and handling the authentication and authorization processes. All these connectors ensure that the data exchange between the Data Marketplace and the Dataspace is governed by the principles of security, privacy, and trust.

The Certificate Authority is responsible for issuing and managing digital certificates, which are used to verify the identities of the participants involved in the data exchange. This ensures the authenticity and integrity of the data and establishes a trusted environment for data transactions.

As mentioned above, the TSG Metadata broker serves as a central repository for publication and discovery of metadata related to data resources available in the Dataspace. It provides functionalities for metadata discovery, retrieval, and access control, allowing users to efficiently search and access relevant data resources. The TSG Metadata broker ensures that metadata is properly organized, standardized, and up to date, promoting interoperability and seamless integration of data from various sources. By facilitating metadata management, the TSG Metadata broker enables the Data Marketplace and other participants to effectively discover and access data resources within the Dataspace. It enhances the overall data exchange process by providing comprehensive information about available datasets, including data formats, access restrictions, quality measures, and other relevant metadata attributes. This allows data consumers to make informed decisions and ensures transparency and traceability in data transactions.

Lastly, the Dynamic Attribute Provisioning Service (DAPS), specifically the mentioned Omejdn framework, provides the necessary protocols and interfaces for secure and controlled access to data. It enforces fine-grained access control policies and enables data providers to define the terms and conditions for sharing their data within the Dataspace ecosystem.

Overall, the integration between the Data Marketplace and the Dataspace through the IDS Testbed (under integration and development at the time of writing) creates a robust and trusted environment for data exchange, ensuring that data transactions are secure, privacy-preserving, and compliant with the defined policies and regulations.



It is important again to note that the integration between the Data Marketplace and the Dataspace is an ongoing development and assessment process. Continuous refinement and evaluation of the integration of these components are necessary to ensure their effectiveness, security and interoperability.

6.5 Blockchain

In this section, we will report the existing implementations which will be used as starting points for the deployment of the infrastructure, and the development of the smart contracts, inside ENERSHARE Task 5.3 of WP5. In the case of one of the two smart contracts here proposed, we have already modified and adapted it for the needs of the project at the time of writing. This version is expected to differ from the next second technological release for few details. In the following sections we assume a basic knowledge about smart contracts and the Ethereum blockchain.

6.5.1 Hardware Infrastructure

Here we describe the infrastructure already used in other projects, which is to be evolved to a next level by the end of the ENERSHARE project.

Basically, the approach used in the past was based on the selection of a public or a private chain according to the requirements and the possible exploitations. The topic has already been introduced in Section 5.7. No choice has currently been made for the ENERSHARE project, thereby being open to discussion according to the emerging needs and requirements, including architectural- and pilot requirements.

6.5.2 Ethereum and public testnets

Ethereum¹⁹ is certainly one of the most famous blockchain networks. It is a decentralized, public blockchain network that was created to enable smart contracts. One reason for its success is its programming language, which, unlike Bitcoin, is Turing complete, and is therefore able to perform any calculation. Additionally, it incorporates a further abstract layer that enables users to design their own transitional rules using smart contracts.

Among the main features of Ethereum is data immutability, which ensures that once a transaction is recorded it cannot be changed or deleted. In addition, all smart contracts are executed automatically and cannot be modified by anyone, which ensures that corruption through them is impossible. Ethereum also uses two types of consensus algorithms, Proof of Work (PoW) and Proof of Stake (PoS), and ensures that there is no downtime, as even if some of the nodes go offline the network continues to function due to the decentralization of the

¹⁹ <https://ethereum.org/it/>



network. Another key feature of Ethereum is Gas, the unit that measures the fee required to successfully conduct a transaction and which allows the network not to be congested by the spam transactions, especially of decentralized applications (Dapps).

Before developing applications on a public mainnet such as Ethereum, public testnets can be used to ensure that Dapps and smart contracts work properly. Developers can use and test the same clients they would use on the public blockchain mainnet, but in testnets they operate on separate ledgers that ensure a low-risk environment in which they can experiment without deployment costs.

In fact, operating on a testnet allows developers to test for vulnerabilities both at an early stage and in later stages to test software updates before they are added to the mainnet. Security is important in these types of applications precisely because of their decentralized structure.

There are many types of testnets, but we are specifically interested in the context of Ethereum-based ones for the exploitation of smart contracts, as it will be described in the next paragraphs.

The main public testnets, used in our past experiences and initiatives, are:

- Kovan²⁰, which uses a Proof of Authority (PoA) model in which only specific nodes are authorized to confirm transactions, developers can only access it using the Parity node software.
- Rinkeby²¹, which uses a PoA consensus mechanism and supports only the Geth software.
- Goerli²², which uses a PoS consensus mechanism to reflect the Ethereum mainnet. Supports Geth, Parity, Nethermind and Hyperledger.
- Sepolia²³, which uses a PoS consensus mechanism to reflect the Ethereum mainnet. Sepolia was designed to simulate harsh network conditions. Compared to other testnets like Goerli, Sepolia total number of tokens is uncapped.

In our experience, we have vastly used Goerli, which was launched as the first native multi-client testnet, efficiently serving client developers and node operators using a simple consensus algorithm. For this reason, it has been perfectly resembled the Ethereum mainnet behaviour, from “The Merge” in September 2022 on²⁴. However, application developers' usage grew exponentially, rendering it less reliable, and this process came to a point where Goerli has been

20 <https://kovan-testnet.github.io/website/>

21 <https://www.rinkeby.io/>

22 <https://goerli.net/>

23 <https://sepolia.dev/>

24 <https://ethereum.org/en/roadmap/merge/>



declared as deprecated²⁵. For this and other reasons we are considering the possibility of moving to Sepolia in the framework of ENERSHARE for all uses cases that involve the usage of a public chain.

6.5.3 Hyperledger and private chains

The problems connected to the usage of the mainnets or public testnets do not arise in the case of private networks, i.e., blockchains locally installed for specific purposes, because on this kind of network, the administrator has full control over the blockchain and can therefore perform all the necessary tests with relatively lower cost- or security risks. The main drawback of the usage of private chains is generally the loss of decentralization of the ownership and administration of the blockchain itself.

Nevertheless, in the case of Business-to-business (B2B) applications, other properties of blockchain are also useful. In enterprise environments, Hyperledger²⁶ is becoming increasingly successful. Hyperledger is an open-source project supported by the Linux foundation and started in 2015 to promote collaborative DLT development. Hyperledger technology, especially for B2B applications, offers unique and interesting features. These include, for example, restrictions on network participation, the ability to define one's own currencies, and various consensus mechanisms. The main distinctions between the features of Ethereum and Hyperledger are shown in the Table below.

Table 36: Comparison between the main features of Ethereum and Hyperledger

Features	Ethereum	Hyperledger
Confidentiality	Public blockchain	Private blockchain
Purpose	Client-side B2C applications	Enterprise-level B2B applications
Governance	Ethereum Developers	Linux Foundation
Participation	Anyone	Organizations having Certificate of Authorization
Programming language	Solidity	Golang ²⁷ , JavaScript ²⁸ , Java ²⁹ , Solidity ³⁰
Consensus Mechanism	PoW/PoS	Pluggable consensus mechanism
Cryptocurrency	Ether	None

²⁵ <https://github.com/eth-clients/goerli>

²⁶ <https://www.hyperledger.org/>

²⁷ <https://go.dev/>

²⁸ <https://developer.mozilla.org/en-US/docs/Web/JavaScript>

²⁹ <https://www.java.com/>

³⁰ <https://github.com/ethereum/solidity>



Speed	Less	More
-------	------	------

There are several projects in Hyperledger to meet different needs that may arise. These include Hyperledger Fabric³¹, Hyperledger Indy³², and many others. Among all of them, Hyperledger Besu seems to be the most promising in the case of B2B applications, as it adds many of the features of Ethereum to the functionality of Hyperledger.

Hyperledger Besu, an open source Ethereum client, was created to combine the benefits of both platforms, especially for commercial applications that use Ethereum. Either the Ethereum public network or private permissioned networks can be used to execute it. Besu comes with a variety of consensus algorithms, such as PoS, PoW, PoA, Istanbul Byzantine Fault Tolerant (IBFT), and Clique, as well as extensive permissioned frameworks created especially for usage in consortium environments.

Hyperledger Besu allows the deployment and execution of smart contracts. In fact, it uses the Ethereum Virtual Machine (EVM) and implements all major consensus algorithms such as PoW, PoS, and PoA. The main features are the ability to add privacy in transactions. This allows the content of transactions to be hidden from un authorized users. Finally, the ability to make networks permissioned allows only authorized nodes to be part of the blockchain network.

For ENERSHARE we are considering the adoption of Hyperledger in all the use cases in which a private chain can be more efficient than a public one, e.g., when the rate of operations on the blockchain become critical for a public mainnet or testnet (e.g., the notarization and/or verification of big amount of data). It is worth noting that legislation is not yet ready for this kind of technology, which is quite new, and private chains are considered to be less reliable, from a bureaucratic point of view, respect to public ones whose immutability is computationally taken for granted.

6.5.4 Smart Contracts

The ability to deploy smart contracts on distributed ledgers (like blockchains) is one of the most intriguing possibilities DLTs provide. Smart contracts are programs designed to carry out tasks when specific criteria are satisfied. Smart contracts enable contracts to be administered between parties without the need for a third party by running as programs on blockchains.

Since smart contracts are executed by code and not by people, there is no possibility of errors during the execution of the smart contract code; not surprisingly, the only possibility of errors concerns the writing of the code by humans, which, however, can be constantly checked by all

³¹ <https://www.hyperledger.org/use/fabric>

³² <https://www.hyperledger.org/use/hyperledger-indy>



network participants due to the decentralized nature of the blockchain. Thanks to smart contracts, it is possible to automate processes that would normally require human involvement. Running them on the blockchain has several advantages, including the fact that it is a decentralized system that exists between all authorized participants, obviating the need for middlemen and reducing time and dispute. Even though traditional methods cannot be matched for speed, cost, or security, DLT and Blockchain enabled applications, such as Decentralized finance (DeFi), Non-Fungible Tokens (NFTs), Decentralized Autonomous Organization (DAO), and other decentralized applications, have all seen a significant increase in the development of smart contracts and their use in recent years.

In ENERSHARE, we will develop smart contracts and related software tool, to leverage the DLT and Blockchain technology for the project purposes. In particular, in the framework of WP5 and the ENERSHARE Marketplace, we have identified two smart contracts, which will come in handy when dealing with transactions and their registration: the ENERSHARE Token, and the Proof-of-Existence (PoE) contracts.

6.5.5 ENERSHARE Token

The first contract regards a new token, which has been developed to implement the “currency” of the Marketplace. It is called **ENERSHARE Token** and its symbol is **EⓈT**, with the “S” circled to resemble the “sharing” concept at the basis of the Project. The smart contract implementing the token is reported below:

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.7.0 <0.9.0;

contract ENERSHARE_Token {
    string public name = "ENERSHARE Token";
    string public symbol = unicode"EⓈT";
    uint256 private totalSupply = 1000000;
    address private owner;
    mapping(address => uint256) balances;
    event Transfer(address indexed _from, address indexed _to, uint256
_value);

    constructor() {
        balances[msg.sender] = totalSupply;
        owner = msg.sender;
    }

    function transfer(address to, uint256 amount) external {
        require(balances[msg.sender] >= amount, "Not enough tokens");
```



```
balances[msg.sender] -= amount;
balances[to] += amount;
emit Transfer(msg.sender, to, amount);
}

function balance(address account) external view returns (uint256) {
    require(msg.sender == account, "Access restricted");
    return balances[account];
}
}
```

In comparison to traditional fiat and new currencies, it is not seen as a cryptocurrency, like the Marketplace is not intended as a place to make necessarily economical transactions. From the conceptual point of view, the **E^ST** can be seen more as a sort of reward for the user, for the data, data services or other services provided, which can be then spent to access data, data services or other services from other users. In this sense, it is more similar to the balance of certain loyalty programs of some airline companies or other ones, than to a currency strictly speaking.

This would encourage the user to spend their tokens and gain their tokens inside the platform itself, since the balance cannot be used outside, like it would have been if we had adopted other cryptocurrencies, like Ether, Bitcoin or IOTA, which could be spent or obtained in other ways also outside the ENERSHARE Marketplace.

In this first early version of the contract, the created token is provided with a fixed total supply of **1000000 E^ST**. This assumption was made to avoid any possible inflationary phenomenon. Of course, with the evolution of the project, this assumption can be revised: e.g., the contract creator can mint new tokens, new tokens can be provided to new accounts, and so on. The contract has been designed also taking into consideration these possible evolutions (taking trace of the total supply and the contract owner) and will be adapted to the emerging needs accordingly.

From the implementational perspective, the source code of the contract is self-explanatory, and thus fully reported above. Basically, it consists of two methods:

- *transfer*: transactional function which allows to transfer a given amount of **E^ST** from the caller to a given address
- *balance*: read-only function which allows to get the amount of **E^ST** belonging to a given address



Since the balance of each user can be maintained private, for the PoLP³³ application, only the owner of the account can see its balance. In its turn, obviously, the transfer function can be done only if the sender's balance is superior to the amount to move.

Different access policies will be possible to be applied in future implementations, and in concert with WP4. In order to obtain an address, a wallet must be created from free tools such as Metamask³⁴, if the contract is intended to be deployed into a public tesnet, such as Goerli or Sepolia, already mentioned in the previous paragraphs.

As next steps, the policy about the provisioning of tokens with the new addresses will be discussed, as well as the possibility of minting new tokens. Technically speaking, also the possibility of evolving the token into a complete ERC20³⁵ compliant token (fungible token) will be discussed, accordingly.

6.5.6 Proof of Existence contract

Another smart contract which has been implemented is the PoE, a contract able to provide a proof and grant the integrity of data provided in the Marketplace. The smart contract is here reported:

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.7.0 <0.9.0;

contract ENERSHARE_ProofOfExistence {
    mapping (bytes32 => string) private proofs;
    event Registered(bytes32);

    function getProof(string calldata data) public pure returns (bytes32) {
        return keccak256(bytes(data));
    }

    function registerData(string calldata data) external returns (bytes32)
    {
        bytes32 hash = getProof(data);
        require(bytes(proofs[hash]).length == 0, "data already registered.");
        proofs[hash] = data;
        emit Registered(hash);
        return hash;
    }
}
```

³³ https://en.wikipedia.org/wiki/Principle_of_least_privilege

³⁴ <https://metamask.io/>

³⁵ <https://eips.ethereum.org/EIPS/eip-20>



```
function getData(bytes32 proof) external view returns (string memory) {
    return proofs[proof];
}

function validateData(string calldata data) external view returns
(bool) {
    return validateProof(getProof(data));
}

function validateProof(bytes32 proof) public view returns (bool) {
    return bytes(proofs[proof]).length > 0;
}
}
```

In this case, we are referring to “data” in the most generic way, not specifically the data that can be shared among the Marketplace users, as defined in the rest of the document. Here “data” means everything which can be “tokenized”, i.e., can be passed to a hash function to obtain a string of fixed length, representative of the input data, as “digest” of the hash function itself. In this sense, “data” can be the data shared among the users, but also the identifier of a transaction, or any document whose integrity must be granted and verified.

In the context of WP5 and the Marketplace, what we intend to register and validate are the identifier of the assets provided within the Marketplace, but also of the transactions happening, as a receipt of the transaction itself.

For this reason and for the sake of simplicity, data are maintained and validated directly on chain to offer a solution completely developed in Ethereum. No private or sensitive data are intended to be registered with this version of the contract. In future developments, we will provide the possibility of uploading directly the hash of the data to be registered (i.e., already encoded data) or even a layer of API to encode the possible private and sensitive data before their notarization.

The contract consists of the two methods plus other three, emerged as necessary for the completeness of the solution:

- *getProof*: pure function which allows to calculate the “proof” of the given data
- *registerData*: transactional function which allows to register the data into the blockchain
- *getData*: read-only function which allows to get the data associated to the given “proof”



- *validateData*: read-only function which allows to validate the authenticity of the given data
- *validateProof*: read-only function which allows to validate the authenticity of the given “proof”

Transactional functions costs gas, while pure and read-only ones do not. Once the user wants to register an asset, data can be registered directly into blockchain by the corresponding transactional function. The user can also get the corresponding “proof”, through the corresponding read-only function. The “proof” can be seen as a fingerprint of the registered data. Technically speaking, it is the digest of the cryptographic algorithm KECCAK 256 (aka SHA-3)³⁶ applied to the encoding of the passed data. The existence of this fingerprint, registered into the blockchain, is an incontrovertible proof of the existence of the corresponding data into the blockchain, and thus the authenticity of those data. In order to do that, the two validation functions can be used, either through the “proof” or directly with the data we can prove as authentic. The corresponding data can be retrieved as well, if registered into the blockchain, by means of the “proof”. This is an important point, since the “proof” can be calculated in a deterministic way (also off-chain), but the corresponding data cannot be retrieved from the “proof”, because the SHA-3, as all the asymmetric cryptographic algorithm, works only in one direction. If and only if a “proof” has been registered in the blockchain, its corresponding data can be retrieved. Otherwise, an empty string is received.

In the following picture, the just depicted scenario is given, from the registration of data to its validation by means of the blockchain.

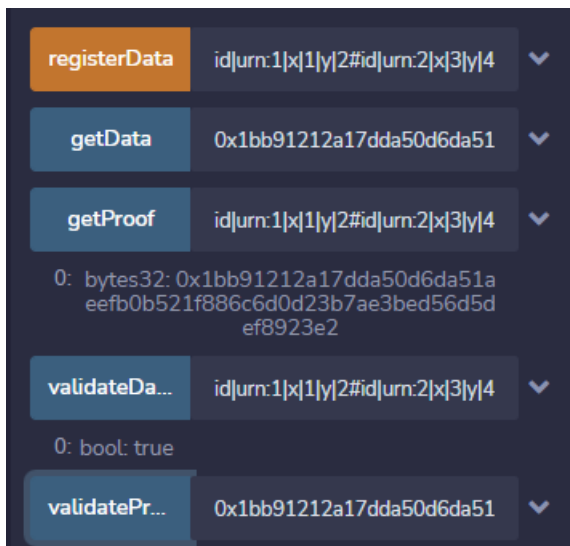


Figure 16: Registration and validation of data by means of the blockchain

³⁶ <https://keccak.team/files/Keccak-implementation-3.2.pdf>



In this example, we have registered some data formatted with the Ultralight 2.0 protocol³⁷ into the blockchain and demonstrated its authenticity and integrity by means of the last two functions. The Ultralight can be easily mapped to JSON, in an intuitive way. For example, the data registered in the example (see the registeredData value, inserted in the first row of the previous screenshot), corresponds to the following JSON array:

Table 37: JSON array of registered data example

Ultralight 2.0	id urn:1 x 1 y 2#id urn:2 x 3 y 4
JSON	<pre>[{ "id": "urn:1", "x": 1, "y": 2 }, { "id": "urn:2", "x": 3, "y": 4 }]</pre>

The contract can be evolved and refined accordingly, e.g., to store data off-chain and register just its proof into the blockchain. As next steps, we intend to evolve the contract according to project needs, e.g., for its integration into the Clearing House as a transaction processor, for the creation of transaction receipts. It is worth mentioning that the development of the PoE contract is being carried out in agreement with WP4, as a joint activity cross-cutting the two deliverables. In any case, the version developed in WP4 can be different to the one presented here for the Marketplace, and in their final shape they may differ also significantly. Data can be maintained on chain or moved off-chain and just the digest of the hash algorithm registered, i.e., its “proof”. All these choices will be done following the project evolution, the single Tasks and pilots needs and the emerging requirements.

³⁷ <https://fiware-iotagent-ul.readthedocs.io/en/latest/usermanual.html>



7 Conclusions

The ENERSHARE project aims to extend the concept of a Data Space, including shared capabilities and services aimed to facilitate and enable the access, sharing, and trading of energy related data assets (datasets, data services) among a variety of data infrastructures and actors.

This Deliverable 5.1 ENERSHARE Data Value Stack (Alpha version) constitutes the preliminary results of the joint activities of WP5, namely Task 5.1 “Publication and data marketplace services”, Task 5.2 “Data usage accounting (Clearing House)”, and Task 5.3 “Tokenised Appstore marketplace and smart contracts for heterogeneous data vs energy services compensation”.

Chapter 2 explained the adopted methodological approach is:

- Definition of actor, roles, responsibilities: An actor is an entity that is a participant in a process and/or a user of a system. A role is a position that an actor has in a process. A responsibility is the set of tasks, activities or actions an actor is expected to perform, or allowed to do, according to the role.
- Writing Use Cases: A Use Case is a definition of a specific business objective that a system needs to accomplish. This is done by describing the interaction the various actors that exist outside of the system have with the system to accomplish the business objective. A template table has been proposed.
- Requirements elicitation and analysis: Functional requirements describe something the system must do. IEEE 29148-2011 standard has been suggested to formalise the requirements.

In Chapter 3, the ENERSHARE Marketplace Use Cases for the different actors have been presented.

In Chapter 4, the high-level functional requirements have been elicited and listed according to a given template.

Chapter 5 presented the software architecture of the ENERSHARE Marketplace and describe the main software components.

Finally, in Chapter 6, this document has provided the description of existing component implementations that will be released as ENERSHARE Data Value Stack Alpha version.



Appendix: Data Monetization and Barter Sharing Incentive Module

Data Contribution

The data Marketplace platform offers clients multiple methods for contributing data, enabling them to choose between manual input via a dedicated Desktop Application or automated data transfers allowing agents to configure their data contribution and management preferences. This application provides an easy-to-use interface for setting up data upload automation and managing account settings.

To ensure that the data is provided on a consistent basis, the platform provides an API that clients can use to automate the upload data at regular intervals. This enables buyers to receive more accurate and reliable forecasts and therefore enhance the data provider rewards. To automate data uploads, sellers can create scripts that pull data from their data sources and upload it to the Marketplace platform’s API. Similarly, buyers can also automate their tools to download forecast data from the platform API.

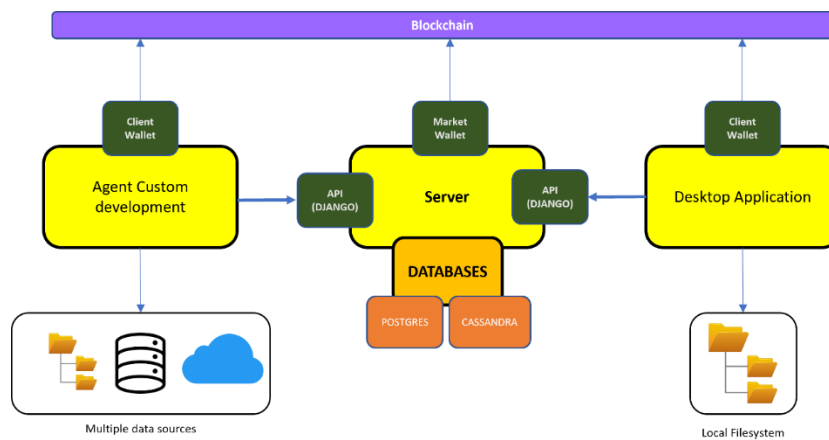


Figure 17: Overview architecture and available interactions for the transfer of data

Data types

The Data Marketplace platform encompasses a diverse range of data types, specifically tailored to meet the needs of the energy domain. These data types encompass energy production time-series related datasets, which serve as the foundation for accurate and reliable forecasting and analysis.



The platform currently supports various energy-related datasets, including solar and wind energy production data. These datasets provide valuable insights into the generation patterns and output levels of energy sources, enabling both agents and market operator to make decisions regarding energy management, grid stability, and resource optimization.

In addition to energy time-series datasets, the Data Marketplace platform is designed to accommodate future expansion and integration of other relevant data types. This includes data such as solar irradiance, wind speed, temperature, and potentially other environmental or operational factors that may impact energy production. The inclusion of these additional data types will further enhance the forecast platform capabilities and enable users to access a comprehensive set of information for advanced forecasting and analysis.

Market Functions

Market Agents Registration

The registration process is a necessary step for users to gain access to the Data Marketplace and utilize its multiple functions. Upon registration, users are required to provide essential information such as their email address, and password. This information serves as the foundation for authenticating and authorizing users' access to the platform.

In addition to the registration process, the Data Marketplace offers the instant creation of a cryptocurrency wallet. The referred wallet provides users with a secure and decentralized storage for their digital tokens, essential to collect revenues or execute transactions within the Market. During the registration process, users are prompted to create a cryptocurrency wallet by setting up a personal identification number (PIN). This PIN acts as an additional layer of security, ensuring that only the authorized user can access and manage their digital tokens in the context of the Data Market or any other transactions they may want to do outside the market (e.g., exchange).

Market Agents Authentication

The authentication function in the Data Marketplace plays a crucial role in verifying users identities and ensuring platform security. Various authentication mechanisms should be implemented to validate users identities before granting them access to the system and enabling their participation in the Market. To access the service, users must have a user account, which can be created through an internal registration process or provided by third-party identity providers common to the dataspace identity provider. This flexibility allows users to choose the most convenient and secure way to authenticate themselves within the service. This authentication process verifies that the provided credentials match the ones associated with the user's account, granting access only to those with valid credentials. To enhance the security of the authentication process, the service also includes a forget password feature and a rate limit access to the authentication process (to prevent abuses). This forgot password



feature is very common and allows users who have forgotten their password to reset it by following a password recovery procedure, which involve an email verification.

Additionally, the Data Marketplace authentication module provides flexibility in terms of wallet integration. While users are assigned a blockchain wallet during registration, they have the option to associate another wallet issued by the same blockchain technology with their user account. This allows users to manage their digital tokens using their preferred wallet, providing them with greater convenience and control over their financial transactions within the platform.

Agents Portfolio Management

The Data Marketplace allows users (agents) to manage their data resources within the platform. It involves the creation, organization, and monitoring of data resources to optimize their value and effectiveness in the Marketplace.

Agents can dynamically manage their portfolio of resources, which may include energy production data, forecasts, and other relevant data sources. From the resource management section, agents can create and define their data resources and assign metadata (name and data upload instructions). They may organize their resources based on different criteria, such as energy source, location, or market sector. By effectively managing their portfolio, agents can showcase their data assets and, in a future development, expand their use to potentially attract buyers or collaborators in a direct data trading environment.

Agents can also monitor the performance and usage of their resources within the Marketplace. This includes tracking the number of data downloads, bids made, and revenue generated from their resources. Agents may also use the information to make informed decisions about their portfolio and optimize their participation in the Marketplace.

Overall, Agents Portfolio Management empowers agents to actively engage with the Data Marketplace, maximize the value of their data resources, and drive collaboration and innovation within the energy forecasting ecosystem.

Agents Market Participation

This function refers to the active involvement of agents in the collaborative activities within the Data Marketplace. Agents can participate in the bidding process, by submitting bids for forecasting services, and potentially engage in the buying and selling of data resources in further developments. For now, agents can only participate in bidding in specific timeframe periods, which are time-limited periods during which the biddings occur. These periods provide agents with opportunities to interact with other participants, submit their bids, and explore potential opportunities to improve their forecasts. When participating in the Marketplace, agents can submit bids for forecasting services. Bids involve specifying the value they are willing to pay for



obtaining potentially more accurate and reliable forecasts for their specific data resources. By participating in the bidding process, agents express their interest and commitment.

Agents participation also encompasses sharing data production resources within the Marketplace. Agents can contribute with their data, allowing other participants to access and use the valuable information they may provide for a Forecast result. Additionally, agents may explore and acquire in the future data resources from other participants to augment their own datasets and improve the forecasting capabilities of their own.

By actively participating in the Marketplace, agents contribute to the collaborative and dynamic nature of the platform, fostering knowledge sharing, innovation, and the development of more accurate and reliable energy forecasts.

Measurements upload

Measurements Upload refers to the process of agents uploading their actual energy production data to the Data Marketplace.

Agents can securely upload their measurements, (e.g., solar or wind energy production data), to contribute to the collaborative forecasting and analysis within the platform and receiving a financial benefit from it.

Agents are responsible for providing accurate and reliable measurements from their energy sources, which are uniquely identified by a resource identifier. This involves collecting data from sensors or monitoring systems that capture real-time or historical production information and can be store or collected in the local filesystem. The uploaded measurements serve as valuable input for the forecasting algorithms and collaborative techniques used within the Data Marketplace forecast operations.

Forecasts download

The download of forecasts function within the Data Marketplace platform enables agents to obtain the forecasts for the energy resources they have listed in their resource settings. The forecasts can be downloaded in various formats, including CSV (Comma-Separated Values) or JSON (JavaScript Object Notation), allowing an easy integration with different applications and analysis tools. Agents may also specify the desired forecast duration and granularity to align with their specific needs and requirements.

Market Engine

The Market Engine serves as the “brain” that orchestrates and manages the various functions and processes within the Data Marketplace related to market sessions, bid validation, data validation, collaborative forecasting, and revenue distribution. It ensures the smooth operation



of the Marketplace, promotes fairness and transparency, and facilitates effective collaboration among agents to maximize the value and benefits derived from the Marketplace ecosystem.

Market Session Management

The Market Engine is responsible for managing market sessions within the Data Marketplace. Market sessions refer to specific time periods during which users can participate in the Marketplace by submitting their bids. The Market Engine handles the scheduling, initiation, and termination of market sessions, ensuring that they occur at predetermined intervals and provide users with the opportunity to engage in trading activities. It also enforces rules and regulations related to session duration, participant eligibility, and other parameters set by the Marketplace administrators.

Agents Bids Validation

During market sessions, agents submit bids for forecasting services access. The Market Engine is responsible for validating these bids (Figure 18). It verifies the authenticity and completeness of bid information, checks for any inconsistencies, or errors, and ensures that bids comply with the Marketplace rules and requirements. The validation process involves verifying the availability of funds in the market wallet with the provided unique transaction identifier provided by the user, confirming the bid value, and checking for any conflicts or overlapping bids. By performing bid validation, the Market Engine ensures a fair and transparent bidding process. There are some important components from the Bid function that we need to explore in detail:

- **Bidding process:** Bids are only allowed during a specific period during which agents can participate in the Marketplace by submitting their bids. The “open” bidding status indicates that the service is accepting bids from users.
- **Resource-based Bidding:** Agents can make bids for each resource they have created in their user account. A resource represents a specific energy source (e.g.: solar farm or a wind farm). Users can submit bids for multiple resources individually, allowing them to participate in the Marketplace for each resource they have configured.
- **Bid Value:** When making a bid, users specify the value they are willing to pay for obtaining the best forecast for their resource. The bid value represents the user's evaluation of the forecast importance and relevance to their energy management needs. Users can set their bid value based on factors such as accuracy requirements and market conditions.
- **Maximum Payment Value:** Along with the bid value, users can define a maximum payment value. This value represents the maximum amount of funds the user is willing to reserve and process as the actual transaction value for the forecast service. The maximum payment value serves as an upper limit, ensuring that the user maintains control over the financial commitment associated with the bid.



- **Reservation and Transaction:** When a user submits a bid, the Data Marketplace system reserves the specified bid value and maximum payment value from the user's wallet.

This reservation ensures that the required funds are set aside for processing the actual transaction once the bid is accepted.

Agent Data Validation

In addition to bid validation, the Market Engine also validates the data shared by agents. Agents may contribute production data from their resources for collaborative forecasting and analysis. The Market Engine checks the integrity, quality, and relevance of the data to ensure its suitability for forecasting purposes. It may perform data cleansing, data normalization, and data consistency checks to eliminate any outliers or inconsistencies that could impact the accuracy of the forecasts. By validating the data, the Market Engine enhances the overall reliability and trustworthiness of the forecasting process.

Collaborative Forecasting

Collaborative forecasting is a function of the Market Engine included in the Data Marketplace Forecast component. It involves leveraging the collective data contributions from multiple agents to generate accurate and reliable forecasts. The Market Engine combines the shared production data, applies advanced forecasting algorithms and techniques, and produces forecasts that reflect the aggregated knowledge and insights of the participating agents. Collaborative forecasting enables agents to benefit from a broader and more diverse dataset, leading to improved forecasting accuracy and enhanced decision-making in energy management.

Note: More insights about this subject and the algorithms employed will be released in the WP7.

Revenue definition and distribution

The Market Engine also plays a role in defining and distributing revenue within the Data Marketplace. It determines the pricing mechanisms, possible fee structures, and revenue sharing models that govern the collaborative data transactions and distribution of buyer's funds. The Market Engine ensures that revenue generated from bid acceptances, data sales, and other Marketplace activities is allocated and distributed to the relevant parties, such as data providers and the Marketplace operators. It enforces the defined revenue distribution rules, tracks financial transactions, and facilitates secure and transparent payment processes.

Note: More insights about this subject and the algorithms employed will be released in the WP7.



Market Wallet

The Market Wallet is a dedicated wallet within the Data Marketplace that serves specific functions related to financial transactions and fund management. It acts as a repository for funds associated with the Marketplace, facilitating secure and transparent handling of financial transactions. The Market Wallet has several key functionalities:

Bid Validation

The Market Wallet is used to validate bids submitted by agents (users) in the Data Marketplace. Agents are required to send funds to the Market Wallet (owned by the Market Operator) as a demonstration of their commitment and financial capability to participate in the bidding process. By sending funds to the Market Wallet, agents provide a transaction identifier that is stored in the intended session and immediately validated by the market engine, granting the agent's bid valid if the unique transaction identifier is available in the Market Wallet and the funds amount correct.

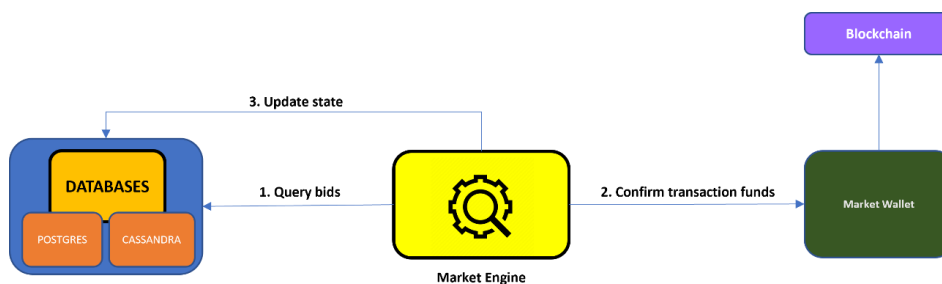


Figure 18: Bid validation process.

Secure Fund Management

The Market Wallet ensures secure management of funds associated with bid validation. It leverages the inherent security features of the underlying technology, such as cryptographic encryption and distributed ledger technology, to safeguard the funds and maintain their integrity throughout the bidding process. This ensures that funds are protected from unauthorized access or tampering.

Transparent Bid Validation

By utilizing the blockchain Market Wallet for bid validation, the Data Marketplace promotes transparency in the bidding process. Participants in the Data Marketplace can verify the existence and funds and transactions in the Market Wallet since the Blockchain technology is open to anyone to explore. This transparency enhances trust and accountability among users, as they can assess the financial commitment of other participants and make informed decisions during the bidding process.



Payment Function

In addition to bid validation, the Market Wallet facilitates the payment function within the Data Marketplace. Once a bid is successfully accepted, and a transaction is finalized, the funds held in the Market Wallet are utilized for payment disbursement to the relevant parties, such as data providers or forecasting service providers. The Market Wallet ensures a streamlined and secure process for transferring funds between buyers and sellers, facilitating smooth financial transactions within the Marketplace.

Integration with Blockchain or Distributed Ledger Technology

The Market Wallet leverages blockchain or distributed ledger technology to ensure the immutability, transparency, and traceability of financial transactions. By integrating with these technologies, the Market Wallet enhances the security and efficiency of funds management and payment processing within the Data Marketplace.

It is important to note that the integration with a blockchain or DLT provides compatibility with various technologies and protocols, ensuring interoperability and facilitating integration with different systems and platforms. This flexibility allows for future scalability and adaptability, enabling the Data Marketplace to leverage advancements in blockchain or DLT technologies and integrate with other blockchain networks or distributed ledger solutions - not limited to financial transactions (incentives).

Market Agent (Desktop application scope)

Local Data Parsers

Local Data Parsers is a component of the Market Agent measurements functionality within the Desktop app scope. These parsers enable the agents to process and interpret the data obtained from various sources in a format that can be understood and utilized by the Market Engine. The Local Data Parsers are responsible for extracting relevant information from the provided datasets, transforming it into a standardized format, and preparing it for further analysis or integration with the Data Marketplace. These parsers present in the APP can handle production and energy datasets, but they can be further developed to parse different types of data, such as weather data, or any other relevant data sources required for forecasting and analysis.

One noteworthy aspect is that the Desktop Application provides users with the ability to customize the Local Data Parsers within the measurement settings section agents can differentiate between "Energy" and "Power" datasets, set the unit value, and define the labels for the value and timestamp in their own datasets. These customizations help the parsers to better understand and process the user's specific data format, ensuring accurate and reliable results.



Local Cryptocurrency Wallet

The Local Cryptocurrency Wallet is a built-in feature within the desktop app that allows Market Agents to securely store and manage their cryptocurrency assets. Cryptocurrency wallets are used to hold the digital tokens required for participating in financial transactions within the Data Marketplace, such as bidding, payment (revenue), or fund management.

The Local Cryptocurrency Wallet provides a convenient and secure solution for Market Agents to store their cryptocurrency assets locally on their devices, ensuring the confidentiality and integrity of their funds.

In summary, the Data Marketplace functions mentioned above, collectively contribute to the creation of a collaborative Data ecosystem. Through the integration of these functions, both data producers (sellers) and consumers (buyers) gain access to a secure platform where they can confidently share and access production data, receive revenues and participate in bidding sessions to acquire more reliable forecasts for their energy resources. These functions not only enable users to actively contribute to the data ecosystem but also foster collaboration, knowledge sharing, and enhanced renewable energy forecasting. By leveraging the power of technology, the Data Marketplace empowers different agents to make data-driven decisions, advance energy management practices, and collectively work towards a sustainable future.

ⁱ https://docs.internationaldataspaces.org/ids-knowledgebase/v/ids-ram-4/layers-of-the-reference-architecture-model/3-layers-of-the-reference-architecture-model/3_5_0_system_layer/3_5_5_clearing_house

