



DOME

(Distributed Open Marketplace for Europe)

D3.1

**DOME Reference Architecture and
Specifications V1**

Project full title

A Distributed Open Marketplace for Europe Cloud and Edge Services

Contract No.

101084071

Strategic Objective

DIGITAL-2021-CLOUD-AI-01-DS-MARKETPLACE-CLOUD

Project Document Number

DOME-D3.1-Vfinal

Project Document Date

7/12/2023

Deliverable Type and Security

R - PU

Main editor

FIWARE Foundation (FF)

Contributors (in alphabetical order)

Alastria, DigitelTS, ED, ENG, e-Group, EXAI, FF, FICODES, IN2, Nicos-Ag, OBS,



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101084071.

Log Table

Version	Date	Change	Author/partner
V0.1	21/03/2023	First draft of document with initial ToC, overall introduction	Juanjo Hierro, FF
V0.2	28/04/2023	Initial contents of chapters 3 and 4	Juanjo Hierro, FF Jesús Ruiz, Alastria Romain Trébuchon, OBS Alexia Germain, OBS Alejandro Nieto, DigitelTS Alejandro Alfonso, DigitelTS Jörg LangKau, Nicos-AG Rajiv Rajani, iSF Dennis Wendland, FF Stefan Wiedemann, FF Oriol Canadés, IN2
V0.3	30/05/2023	Further refinement of chapters 3 and 4. Initial contents of Appendix I	Previous editors plus: Antoine Maisonneuve, OBS Bruno Salinier, OBS Paolo Fabriani, ENG Tim Smyth, FF Beknazar Esenbek, FF
V0.4	31/05/2023	Design of structure for chapter 5	Paolo Fabriani, ENG
V0.5	30/06/2023	Chapter 4: DOME Persistence Layer	Stefan Wiedemann, FF Tim Smyth, FF Oriol Canadés, IN2
		Further refinement of chapters 3, 4 and Appendix I. Initial contents of chapter 5. Initial contents of Appendix II and III.	Previous editors plus: Marialuna de Tommaso, ENG Luca Santanché, ENG Vincenzo Masucci, EXAI Giuseppe Cafiso, DHUB Giovanni Frattini, ENG Alfonso Pietropaolo, ENG Bruno Salinier, OBS Panos Bouklis, ED



			Kostas Panagopoulos, ED Francisco de la Vega, FICODES
V0.6	31/07/2023	Chapter 3: DOME Trust and IAM Framework, Appendixes I, II and III	Jesús Ruiz, Alastria Alejandro Nieto, DigitelTS Romain Trébuchon, OBS
		Further refinement of chapters 3, 4, 5 and Appendix I, II and III	Previous editors plus: Maria Serafina Cefarelli, EXAI Csaba Kormoczi, e-Group
V0.7	01/09/2023	Further refinement of chapters 3, 4, 5 and Appendix I, II and III. First consolidated version for peer review	Previous editors
		Chapter 5: DOME Marketplace Features	Paolo Fabriani, ENG Francisco de la Vega, FICODES Csaba Kormoczi, e-Group
V0.8	04/09/2023	Addition of Overview chapter 2.	Hierro, FF
		Peer-review. Adding items to list of acronyms and related documents and resources.	Chief editors plus Alexia Germain, OBS Giuseppe Cafiso, DHUB Giovanni Frattini, ENG
V0.9	05/10/2023	Review of chapter 3, clean version	Alexia Germain, OBS
V1.0	21/11/2023	Final formatting check, and figures numbering	Laura Pucci, ENG
VFinal	7/12/2023	Included tables and removed reference to external docs in Chp 5, submission to EC	Laura Pucci, ENG



Table of Contents

1	Introduction	20
1.1	Executive Summary	20
1.2	Intended audience	20
1.3	Structure of the document	20
1.4	Related documents and resources	21
2	DOME General Overview	22
2.1	Overview of roles in DOME.....	22
2.1.1	Cloud and edge service providers	23
2.1.2	Federated Marketplaces.....	24
2.1.3	Customers.....	25
2.1.4	Operators of the DOME technical infrastructure	25
2.1.5	Third parties integrating and offering complementary services	25
2.1.6	Members of governance and supervisory bodies	26
2.2	Overall technical approach	26
2.2.1	Basic Information model.....	26
2.2.2	DOME Shared Catalogue and Transactions Ledger (Distributed Persistence Layer)	28
2.3	Journey of DOME users	29
2.3.1	Service providers journey.....	29
2.3.2	Customers journey	38
3	DOME Trust and IAM Framework	39
3.1	Trust Framework	39
3.1.1	ID Binding	39
3.1.1.1	Cross-border use of mutually recognised electronic identification means	40
3.1.1.2	ID Binding and the Verifiable Credential	41
3.1.1.3	About identifiers for legal persons.....	43
3.1.1.4	About identifiers for natural persons	45
3.1.1.5	Disclaimer.....	45
3.1.2	Proof of participation	46
3.1.3	Proof of Issuing Authority	47
3.1.3.1	Trusted Issuers Registries	47
3.1.3.2	The List of Trusted Lists (LOTL)	48



3.2 Onboarding of DOME participants	49
3.2.1 Identification and Verification of future participants.....	49
3.2.1.1 Legal basis	50
3.2.1.2 The LEARCredential.....	54
3.2.1.3 The identification and verification phase	55
3.2.2 Registration of new participants in the Trusted Lists (DigitelTS)	55
3.2.2.1 Root TAO registration	58
3.2.2.2 TAO registration	59
3.2.2.3 Domain-specific Trusted Issuer.....	60
3.2.2.4 Trusted Issuers registration	61
3.2.2.5 Trusted Participant Registrar registration.....	62
3.2.2.6 Registering of a Trusted Issuer of a domain using Trusted List(s).....	63
3.2.2.7 Registering of a Trusted Issuer of a domain using a trust chain.....	64
3.3 Authentication and Authorization with Verifiable Credentials.....	65
3.3.1 Introduction	65
3.3.1.1 Signature of the Verifiable Credentials.....	69
3.3.1.2 A taxonomy of Verifiable Credentials	70
3.3.1.3 Authentication requires a VerifiableID	71
3.3.1.4 Access Control requires a Verifiable Authorisation	72
3.3.1.5 Combining VerifiableID and Verifiable Authorisation in the LEARCredential .	73
3.3.1.6 The LEARCredential for some identification processes	75
3.3.2 The LEARCredential through an example	76
3.3.2.1 Claims identifying the employee	76
3.3.2.2 DID of the employee	79
3.3.2.3 legalRepresentative	80
3.3.2.4 rolesAndDuties of the LEAR	81
3.3.2.5 Assembling the pieces together	84
3.3.3 Issuing the LEARCredential	85
3.3.3.1 Overview	85
3.3.3.2 Authentication.....	87
3.3.3.3 Credential Offer	88
3.3.3.3.1 Credential Offer Parameters.....	89
3.3.3.3.2 Contents of the QR code.....	90
3.3.3.4 Credential Issuer Metadata.....	91
3.3.3.4.1 Credential Issuer Metadata Parameters	92

3.3.3.5 OAuth 2.0 Authorization Server Metadata.....	93
3.3.3.6 Access Token.....	93
3.3.3.6.1 Token Request	94
3.3.3.6.2 Successful Token Response	94
3.3.3.6.3 Token Error Response.....	95
3.3.3.7 Request and receive Credential.....	95
3.3.3.7.1 Binding the Issued Credential to the identifier of the End-User possessing that Credential	96
3.3.3.7.2 Credential Request	96
3.3.3.7.3 Credential Response.....	98
3.3.3.7.4 Credential Error Response	98
3.3.3.7.5 Credential Issuer Provided Nonce	99
3.3.4 Authenticating with Verifiable Credentials	99
3.3.4.1 Overview	99
3.3.4.2 Starting the OpenID for Verifiable Presentations flow	102
3.3.4.3 Generating the Authorization Request	104
3.3.4.4 Verification of the Authorization Request	107
3.3.4.5 Creating and sending the Authorization Response	108
3.3.4.6 Authenticating the user with the LEARCredential.....	110
3.3.5 Access control with Verifiable Credentials	113
3.3.5.1 Overview	113
3.3.5.2 Determine the Authorization Policies which apply to the request	115
3.3.5.3 Determine which Trusted Issuer Lists should be queried	118
3.3.5.4 Query the Trusted Issuer Lists	119
3.3.5.5 Compute the decision and allow (or not) access.....	120
3.3.5.6 Summary of the authorization process.....	121
3.4 Detailed workflows illustrating interactions among components based on a reference use case.....	122
3.4.1 Access to DOME functions by DOME users.....	122
3.4.1.1 Explaining the Reference Use Case	122
3.4.1.1.2 GoodAir: the company that will perform the onboarding process	123
3.4.1.1.3 Jesus Ruiz: COO (Chief Operating Officer) of GoodAir	123
3.4.1.1.4 RealTruth: a TSP (Trust Service Provider).....	124
3.4.1.1.5 John Doe: an administrative employee of GoodAir	126
3.4.1.2 Onboarding of organisations in DOME.....	127

3.4.2 Use of the DOME Trust and IAM framework by data/app service providers (Use case: Air Quality).....	127
3.4.2.1 Description of roles of the Air Quality Monitoring app.....	127
3.4.2.2 Acquisition of rights to use Air Quality Monitoring app by customer means that customer becomes Trusted issuer of that app specific VCs.....	127
3.4.2.3 Operations by Admin as well as End users and implication in access.....	134
3.4.3 Use of the DOME Trust and IAM framework by data/app service providers (Use Case: Packet Delivery).....	143
3.4.3.1 Parties involved	144
3.4.3.1.1 Data Service Provider: Packet Delivery Company	145
3.4.3.1.2 Data Service Consumer: Happy Pets Inc.	147
3.4.3.1.3 Data Service Consumer: No Cheaper Ltd.	147
3.4.3.1.4 Trust Anchor Framework	148
3.4.3.2 Verifiable Credentials in the ecosystem	151
3.4.3.2.1 Employee of Packet Delivery.....	151
3.4.3.2.2 Employee of Happy Pets (or No Cheaper)	155
3.4.3.2.3 Customer of Happy Pets (or No Cheaper).....	157
3.4.3.2.4 Role-based access.....	158
3.4.3.3 Detailed workflows.....	158
3.4.3.3.1 Create Offering.....	158
3.4.3.3.2 Acquisition of rights / Activation	167
3.4.3.3.3 Access to data service.....	177
3.4.3.3.4 Issuing tokens for Connectors / application context.....	188
4 DOME Marketplace Persistence Layer.....	191
4.1 Introduction.....	191
4.1.1 Document Objective	191
4.1.2 Context and Rationale for Adopting a Decentralised and Federated Data Architecture in the DOME Solution Ecosystem	191
4.2 Theoretical Foundations	192
4.2.1 Decentralised Persistent Layer.....	192
4.2.1.1 Definition	192
4.2.1.2 Decentralization in Data Storage	192
4.2.1.3 Blockchain and Context Broker as a Technology Solution	193
4.2.1.3.1 Blockchain Technology for Distributed Persistence	193
4.2.1.3.2 Context Broker as an Off-chain Persistence Solution	194
4.2.1.4 Decentralised Networks.....	194



4.2.1.4.1 Network Nodes and Distributed Storage	194
4.2.1.4.2 Replication and Consensus Mechanisms	195
4.2.1.4.3 Data Security and Encryption	199
4.3 Proposed Architecture	199
4.3.1 Overview of the Architecture	199
4.3.1.1 Blockchain Federation Architecture	201
4.3.2 Components of the DOME Decentralised Persistence Layer	202
4.3.2.1 TM Forum APIs	202
4.3.2.2 Storage Back-End	202
4.3.2.2.1 Standardised interfaces and mechanisms	204
4.3.2.2.2 CEF Context Broker reference implementation.	204
4.3.2.2.3 Connecting Storage Backend and Blockchain Connector.....	205
4.3.2.3 Blockchain Connector.....	205
4.3.2.3.1 Main Functionalities of the Blockchain Connector	205
4.3.2.3.2 APIs and Interfaces of the Blockchain connector.....	207
4.3.2.3.3 Considerations about Blockchain Connector	209
4.3.2.3.4 Creating blockchain Events	209
4.3.2.3.5 Publishing of Event Logs.....	209
4.3.2.3.6 Subscribing to Event Logs.....	209
4.3.2.3.7 Retrieving data of the origin Storage Back-End from resolving a blockchain event.....	210
4.3.2.4 Blockchain Abstraction Interface.....	210
4.3.2.7 Interchain Event Distributor (IED)	211
4.4 Data Federation.....	211
4.4.1 The model of transferring data in a blockchain.	212
4.4.2 The Data Flow.....	212
4.5 Data considerations	216
4.5.1 The Blockchain Event	216
4.5.1.1 Safe pointer to the origin data in the blockchain transaction: the Hashlink. .	217
4.5.1.2 Type of the Entity: the Metadata	217
4.5.2 Security considerations about Events or Logs in Blockchain transactions.	217
4.5.2.1 Data Integrity	217
4.5.2.2 Information Leakage	218
4.5.2.3 Auditing	218
4.5.2.4 Policy Distribution	218

4.6 Implementation Considerations	219
4.6.1 Joining the Marketplace	219
5 DOME marketplace features	221
5.1 Background	221
5.1.1 Concepts and Terminology	221
5.1.2 The procurement process	223
5.2 Subscription Management	225
5.3 Catalogue Management	225
5.3.1 Managing Specifications for Products, Services and Resources	226
5.3.2 Managing Categories	227
5.3.2.1 Categories Lifecycle States	228
5.3.2.2 Creating categories.....	229
5.3.2.3 Retrieving categories	229
5.3.2.4 Updating categories.....	229
5.3.2.5 Deleting categories	229
5.3.3 Managing Offerings	230
5.3.3.1 Offerings lifecycle	230
5.3.3.2 Offerings management in the DOME central marketplace	231
5.3.3.3 Agreements management.....	232
5.3.3.4 Revenue sharing policy definition	232
5.3.4 Browsing the DOME central marketplace catalogue.....	233
5.3.5 Replication and visibility of catalogues in the federation	233
5.3.5.1 Establishment of replication and visibility policies	234
5.3.5.2 Establishment of acceptance rules on federated target marketplaces.....	234
5.3.5.3 How policies are disseminated in the ecosystem	234
5.3.5.4 Enforcement of policies	235
5.4 Service Brokerage	236
5.4.1 Overview	236
5.4.2 Order Placement.....	236
5.4.2.1 Federated Scenarios for order placement.....	237
5.4.2.1.1 Placing orders on DOME for offerings linked to a Federated Marketplace	237
5.4.2.1.2 Placing orders on DOME for offerings linked to the DOME marketplace	239
5.4.2.1.3 Placing orders on Federated Marketplaces	240

5.4.2.2 Placing orders on the DOME marketplace	240
5.4.2.2.1 The product order information model.....	240
5.4.2.2.2 Configuring the product offering	243
5.4.2.2.3 Agreement customisation	243
5.4.2.2.4 Including customer and billing information.....	244
5.4.2.2.5 Submitting the order to the Decentralised Persistence Layer	244
5.4.3 Service Provisioning.....	244
5.4.3.1 Provisioning scenarios in the DOME Federated environment	245
5.4.3.1.1 Federated Marketplace and Provider linked with a legacy integration ..	246
5.4.3.1.2 Federated Marketplace and Provider linked through DOME.....	247
5.4.3.1.3 Different Selling and Providing Marketplaces	249
5.4.3.2 The Provisioning Subsystem of the DOME marketplace	251
5.4.3.2.1 The DOME-compliant Provisioning Subsystem	251
5.4.3.2.2 The Plugin-based Provisioning Subsystem	252
5.4.3.2.3 The Product/Service/Resource Inventory Subsystem.....	252
5.4.4 Consumption and Usage Tracking	253
5.4.4.1 Overview	253
5.4.4.2 Usage Tracking in DOME	254
5.4.5 Billing	256
5.4.5.1 Overview	256
5.4.5.2 Billing-related aspects of the DOME ecosystem	257
5.4.5.3 Overall approach	257
5.4.5.4 A deeper analysis of the Billing feature.....	258
5.4.5.4.1 Scheduling	259
5.4.5.4.2 Periodicity	259
5.4.5.4.3 The time series database	260
5.4.5.4.4 Impact on revenue sharing.....	260
5.4.5.5 Billing scenarios in the DOME federated environment	260
5.4.6 Revenue Sharing	265
5.4.6.1 Overview	266
5.4.6.2 The reselling process.....	266
5.4.6.3 The integration of the reseller process in the DOME ecosystem.....	267
5.4.6.4 Relationship with the billing process	268
5.4.6.5 Relationship with the invoicing process.....	268
5.4.7 Invoicing.....	268

5.4.7.1 Overview	268
5.4.7.2 What is an Invoice in the DOME Marketplace?	269
5.4.8 Payment.....	269
5.4.8.1 Main technical overview.....	270
5.4.8.2 Main scenarios	270
5.4.8.2.1 Payment with new card, with SCA (Strong Customer Authentication)...	270
5.4.8.2.2 Payment with existing card, without SCA	270
5.4.8.2.3 Payment with existing card, with SCA	271
5.4.8.2.4 Bank card data update during payment because of expiration	271
5.4.8.2.5 Alternative bank card selection during payment because of insufficient funds, without SCA	271
5.4.8.3 State diagram	273
5.4.8.4 Main concepts	274
5.4.8.5 Centralised payment gateway accounts.....	276
5.4.8.5.1 Payout.....	276
5.4.8.6 Decentralised payment gateway accounts.....	277
5.4.8.7 Payment with bank transfer	277
5.4.8.8 Interactive payments.....	277
5.4.8.9 Recurring / non interactive payments.....	277
5.4.8.10 Backend services.....	278
5.4.8.10.1 Payment processor	278
5.4.8.10.2 Card processor.....	278
5.4.8.10.3 Audit logger	278
5.4.8.10.4 Customer handler.....	278
5.4.8.11 Example payment flows	278
5.4.8.11.1 Interactive flow	278
5.4.8.11.2 Non-interactive flow	280
5.5 Indexing and Search.....	280
5.5.1 Indexing subsystem	283
5.5.1.1 Indexing a new Item.....	284
5.5.1.2 Indexing a Catalogue.....	285
5.5.2 Search subsystem.....	285
5.5.3 NLAPI Services	287
5.5.3.1 Document Analysis	288
5.5.3.2 Document Classification	290

5.5.3.3 NLAPI Use Example	293
5.6 Monitoring, Reporting and Analytics	297
5.6.1 Description of interactions with other components.....	298
5.6.2 Creating the monitoring dashboards.....	299
5.6.2.1 Data source	299
5.6.2.2 Dataset	299
5.6.2.3 Data visualisation	300
5.6.2.4 Monitoring dashboard	301
5.6.3 Accessing monitoring dashboards.....	302
5.6.3.1 View and edit a monitoring dashboard.....	302
5.6.3.2 Managing permissions and sharing	303
5.6.4 Importing, exporting and APIs	306
5.6.4.1 Import and export via web UI	306
5.6.4.2 Monitoring dashboard-specific API	307
5.6.5 Preconfigured KPIs	308
5.6.5.1 All	308
5.6.5.2 Service consumer	309
5.6.5.3 Service provider.....	309
5.6.5.4 Federated marketplace provider	309
5.6.5.5 DOME operator	309
5.7 User Support	310
5.7.1 General Concepts	310
5.7.1.1 Focus of the customer service	310
5.7.1.2 Roles/people involved in customer support.....	311
5.7.1.3 Language	311
5.7.1.4 Managed Data	311
5.7.2 Customer support technical platform	312
5.7.2.1 Knowledge base	312
5.7.2.2 Chatbot.....	312
5.7.2.2.1 Input data	314
5.7.2.2.2 Provided functionalities	314
5.7.2.3 Troubleticketing system	314
5.7.3 Service level and KPIs	316
5.7.3.1 Service Level Agreement.....	316
5.7.4 Sample Use case	317

5.7.4.1 Query categories	317
5.7.4.2 Query samples	319
6 Appendix I: example of onboarding of organisations in DOME	328
6.1 Participants.....	328
6.1.1 SmartCityDS: an instance of a Data Space for Smart Cities.....	329
6.1.2 GoodAir: the company that wants to will perform the onboarding process	329
Jesus Ruiz COO (Chief Operating Officer) of GoodAir.....	329
6.1.4 RealTruth: a TSP (Trust Service Provider)	330
6.1.5 John Doe: an administrative employee of GoodAir.....	332
6.2 The LEARCredential.....	333
6.2.1 Claims identifying the employee.....	333
6.2.2 DID of the employee	335
□6.2.3 legalRepresentative.....	336
6.2.4 rolesAndDuties of the LEAR.....	337
6.2.5 Assembling the pieces together	339
6.3 Issuing the LEARCredential.....	340
6.3.1 Overview.....	340
6.3.2 Authentication	343
6.3.3 Credential Offer.....	344
6.3.3.1 Credential Offer Parameters	345
□6.3.3.2 Contents of the QR code.....	346
6.3.4 Credential Issuer Metadata	347
6.3.4.1 Credential Issuer Metadata Parameters.....	348
6.3.5 OAuth 2.0 Authorization Server Metadata	348
6.3.6 Access Token	349
6.3.6.1 Token Request	349
□6.3.6.2 Successful Token Response.....	350
□6.3.6.3 Token Error Response	351
6.3.7 Request and receive Credential	352
6.3.7.1 Binding the Issued Credential to the identifier of the End-User possessing that Credential.....	353
6.3.7.2 Credential Request.....	353
6.3.7.2.1 Proof Type.....	353
□6.3.7.3 Credential Response.....	354
□6.3.7.4 Credential Error Response	355

6.3.7.5 Credential Issuer ProvidedNonce	355
7 Appendix II: JAdES signatures of Verifiable Credentials	357
7.1 Introduction.....	357
7.2 Choosing between Qualified and Advanced Signatures.....	357
7.3 Signing using JAdES	358
7.3.1 Timestamping of JWTs.....	358
7.3.2 Example of a JWT Credential signed using JAdES	358
8 Appendix III: remote Digital Signature Service (rDSS).....	362
8.1 Introduction.....	362
8.2 JAdES cloud signature related endpoints	362
8.2.1 Service authorization and authentication	363
8.2.2 Credential authorisation for remote signatures	363



List of figures

Figure 1 - High-level vision of DOME architecture, operating model and roles	23
Figure 2 - TM Forum information model applied in DOME	27
Figure 3 - High-level architecture of the DOME Distributed Persistence Layer	28
Figure 4 - Service providers journey in DOME	29
Figure 5 - interactions among components during subscribe stage	31
Figure 6 - interactions among components during reference stage	33
Figure 7 - interactions among components during Product Ordering	34
Figure 8 - Lifecycle of Product Orders	35
Figure 9 - Lifecycle of a Product (instance)	36
Figure 10 - Interactions during the lifecycle of Product instances	37
Figure 11 - Consumers journey in DOME	38
Figure 12 - Classes of credentials	71
Figure 13 - Figure High level view of issuance of LEARCredential	75
Figure 14 - Figure LEAR subject identification data	78
Figure 15 - Figure LEAR roles and duties	82
Figure 16 - Figure Overview of participants	85
Figure 17 - Figure Issuance authentication	87
Figure 18 - Figure Credential offer	88
Figure 19 - Figure Issuer metadata	92
Figure 20 - Figure Access Token	93
Figure 21 - Figure Request and receive Credential	95
Figure 22 - Figure Architecture overview	100
Figure 23 - Figure Starting the authentication phase	103
Figure 24 - Figure QR code with the example Authorization Request endpoint inside	104
Figure 25 - Figure Starting the authentication phase	107
Figure 26 - Figure Starting the authentication phase	108
Figure 27 - Figure Starting the authentication phase	111
Figure 28 - Figure High level Authentication logical architecture	115
Figure 29 - Figure Determine the Authorization Policies	115
Figure 30 - Figure Determine Trusted Issuer Lists to query	119
Figure 31 - Figure Query the Trusted Issuer Lists	120
Figure 32 - Figure Compute the decision	121
Figure 33 - Figure Summary of the Authorization process	122
Figure 34 - SmartCityDS: an instance of a Data Space for Smart Cities	123



Figure 35 - Overall architecture	144
Figure 36 - Sequence diagram for step “Create Offering”.....	160
Figure 37 - Sequence diagram for step “Acquisition of Rights / Activation”.....	170
Figure 38 - Architecture diagram for step “Change PTA by Happy Pets customer”	178
Figure 39 - Sequence diagram for step “Change PTA by Happy Pets customer”	179
Figure 40 - The procurement process	224
Figure 41 - Managing resource, service and product specifications	227
Figure 42 - Lifecycle of categories.....	228
Figure 43 - Lifecycle of offerings	231
Figure 44 - Ordering: relational scenario	238
Figure 45 - Ordering: transactional scenario	239
Figure 46 - Ordering: products that are local to the DOME marketplace.....	240
Figure 47 - Lifecycle of a product order	242
Figure 48 - Provisioning: general scenario	245
Figure 49 - Provisioning: legacy integration between Marketplace and Provider	246
Figure 50 - Provisioning: legacy integration between Marketplace and Provider	247
Figure 51 - DOME-based interaction between marketplace and provider.....	248
Figure 52 - Provisioning: multiple marketplaces and legacy integration between marketplace and provider.....	249
Figure 53 - Multiple marketplaces and DOME-based interaction with the provider.....	250
Figure 54 - Multiple marketplaces and DOME-based interaction with the provider.....	251
Figure 55 - Usage tracking: general scenario.....	255
Figure 56 - Usage tracking: general scenario with legacy integration between Marketplace and Provider.	256
Figure 57 - Billing: single marketplace, flat rate	261
Figure 58 - Billing: single marketplace, pay-per-use.....	263
Figure 59 - Billing: product sold by a marketplace, billed by another one.	265
Figure 60 - Search and Indexing: architecture.....	282
Figure 61 - Indexing a new item	284
Figure 62 - Indexing a catalogue	285
Figure 63 - Search workflow.....	286
Figure 64 - NLAPI Services Architecture	288
Figure 65 - Document Analysis Service JSON IN.....	288
Figure 66 - Document Analysis Service JSON IN.....	290
Figure 67 - Document Classification Service JSON IN	291

Figure 68 - Document Classification Service JSON OUT	291
Figure 69 - Document Classification Service JSON OUT Categories	292
Figure 70 - NLAPI Use Example	293
Figure 71 - Dropbox Main Lemmas	294
Figure 72 - Dropbox Main Sentences	294
Figure 73 - Dropbox Main Phrases	295
Figure 74 - Dropbox Main Symcons	295
Figure 75 - Dropbox Entities	296
Figure 76 - Dropbox IT Category	296
Figure 77 - Dropbox IT Category	297
Figure 78 - Main interactions of Monitoring Dashboards with other DOME components ...	298
Figure 79 - Adding a data source (DOME administrator only)	299
Figure 80 - Adding a dataset out of a data source	300
Figure 81 - Examples of available data visualisation chart types	300
Figure 82 - Configuring the data visualisation chart	301
Figure 83 - Creating a new monitoring dashboard	301
Figure 84 - Adding a chart in the dashboard by drag&drop	302
Figure 85 - Modifying charts in a dashboard and adding graphical elements	302
Figure 86 - List of accessible dashboards	303
Figure 87 - Start editing an existing dashboard	303
Figure 88 Part of permissions that the DOME administrator can assign to users and user groups.....	304
Figure 89 - A user can be granted access to a dataset (1), chart (2) and dashboard (3) by adding them in the 'OWNERS' fields	306
Figure 90 - Importing data through .csv, excel or columnar file	307
Figure 91 - Export options for a chart	307
Figure 92 - Web page with all the available API calls, instructions on calls and responses	308
Figure 93 - Figure LEAR subject identification data	334
Figure 94 - Figure LEAR roles and duties	338
Figure 95 - Figure Overview of participants	341
Figure 96 - Figure Issuance authentication	343
Figure 97 - Figure Credential offer	344
Figure 98 - Figure Issuer metadata	347
Figure 99 - Figure Access Token	349
Figure 100 - Figure Request and receive Credential	352

Acronyms

Acronym	Definition
API	Application Programming Interface
DID	Decentralized IDentifier
DCAT	Data Catalog
JAdES	JSON Advanced Electronic Signatures
JWS	JSON Web Signature
JWT	JSON Web Token
ODRL	Open Digital Rights Language
OID4VP	OpenID Connect for Verifiable Presentations
OID4VCI	OpenID Connect for Verifiable Credentials Issuance
PEP	Policy Enforcement Point
PIP	Policy Information Point
PDP	Policy Decision Point
PRP/PAP	Policy Registry Point / Policy Administration Point
SIOPv2	Self-Issued OpenID Provider version 2
URI	Uniform Resource Identifier
VC / VP	Verifiable Credential / Verifiable Presentation



1 Introduction

1.1 Executive Summary

Cloud computing is identified as a central piece of Europe's digital future, giving European businesses and public organisations the data processing technology required to support their digital transformation. The European Commission thereby stepped up its efforts to support cloud uptake in Europe as part of its strategy, notably with the pledge to facilitate "the set-up of a cloud services marketplace for EU users from the private and public sector". DOME will materialise the envisioned online marketplace, providing the means for accessing trusted services, notably cloud and edge services, building blocks deployed under the Common Services Platform and more generally any software and data processing services developed under EU programmes such as the Digital Europe Programme, Horizon 2020 or Horizon Europe. Relying on Gaia-X concepts and open standards, DOME will provide the finishing touch to the technical building that the Digital Europe Program is creating for boosting the development and adoption of trusted Cloud and Edge services in Europe. It will provide the single point for enabling customers and service providers to meet each other in a trustful manner. DOME will take the form of a federated collection of marketplaces connected to a shared digital catalogue of cloud and edge services. Each of the federated marketplaces will be independent or connected to the offering of a given cloud providers which, in turn, can be classified as cloud IaaS providers or cloud platform providers (each of which provide a platform targeted to solve the integration of vertical data/application services from a given vertical domain, like smart cities or smart farming, or the integration of certain type of data/application services, e.g., AI services). DOME will rely on the adoption of common open standards for the description of cloud and edge services and service offerings as well as their access through a shared catalogue.

This document describes the Reference Architecture and Detailed Technical Specifications of the DOME components.

1.2 Intended audience

DOME partners involved in technical activities and DOME reviewers.

1.3 Structure of the document

This document is divided into 4 chapters:

Chapter 1: Introduction - provides a summary of this document



Chapter 2: DOME General Overview - describes the overall structure of the project, including different roles of users, the overall technical approach and the user journeys.

Chapter 3: DOME Trust and IAM Framework - provides a description of the decentralised Trust and Identity and Access Management (IAM) framework which are implemented in DOME.

Chapter 4: DOME Marketplace Persistence Layer - provides a description of the DOME Persistence Layer, focusing on the specific goals and priorities associated with its implementation, including the Shared Catalog and the Transactions Layer.

Chapter 5: DOME marketplace features - provides a description of the DOME marketplace, including the design of the processes involved in a federated procurement scenario and high-level architecture of the various subsystems delivering such features.

Chapter 6: Appendix I: example of onboarding of organisations in DOME - provides examples with detailed technical descriptions of some user journeys in DOME..

Chapter 7: Appendix II: JAdES signatures of Verifiable Credentials - provides a description of the approach to digital signatures of Verifiable Credentials in DOME, to achieve high legal certainty and alignment with the EU eIDAS (and upcoming eIDAS2) regulatory framework..

Chapter 8: Appendix III: remote Digital Signature Service (rDSS) - provides a description of the approach to remote/cloud digital signatures, enabling the use of advanced and qualified signatures for Verifiable Credentials and other types of documents, increasing legal certainty and making Verifiable Credentials equivalent to handwritten signed documents.

1.4 Related documents and resources

Following is a list of valuable links to relevant documents and resources:

- [DOME Technical Overview presentation](#)
- [TM Forum APIs \(summary by FIWARE Foundation\)](#)
- [EU Digital Identity Wallet Architecture and Reference Framework](#)
- [DSBA Technology Convergence: Discussion Document](#)
- [Digital Signature Service - DSS](#)
- [DID ETSI Legal person Semantic Identifier Method Specification \(did:elsi\)](#)



2 DOME General Overview

DOME will take the form of a **federated collection of marketplaces connected to a shared digital catalogue of cloud and edge services**. These services can be further classified as:

- data services, providing access to data
- application (app) services, which gather and process data, and typically deliver data results
- cloud or edge infrastructure services, supporting the deployment and execution of data/app services

Cloud and edge infrastructure service providers, in turn, can be classified as cloud/edge IaaS providers or cloud/edge Platform service providers (in this latter case, providing a platform targeted to solve either the integration of several data/app services linked to a given application domain, like smart cities or smart farming, or the integration of certain type of data/app services, e.g., AI services)

Each of the federated marketplaces in DOME will be a marketplace provided by an independent marketplace provider or a marketplace connected to the offering of a given cloud / edge infrastructure service provider (IaaS or platform service providers). Besides these marketplaces, DOME implements all the functionalities of a marketplace itself, including a marketplace portal through which cloud/edge service providers may register their product offerings and end customers can procure offered products.

DOME will rely on the adoption of common open standards for the description of cloud and edge services and service offerings as well as their access through a shared catalogue.

The following subsections elaborate on the roles that organisations can play with respect to DOME as well as some details of the technical architecture.

2.1 Overview of roles in DOME

Six different roles can be played by organisations involved in the ecosystem created around DOME as illustrated in figure 1: cloud and edge service providers, marketplace providers, customers, the operators of the DOME technical infrastructure, third parties capable of integrating and offering their services complementing those implemented in the DOME technical infrastructure, and members of governance and supervisory bodies.

The following subsections will introduce the mentioned roles. Governance and Supervision bodies will be described when tackling Objective #10.



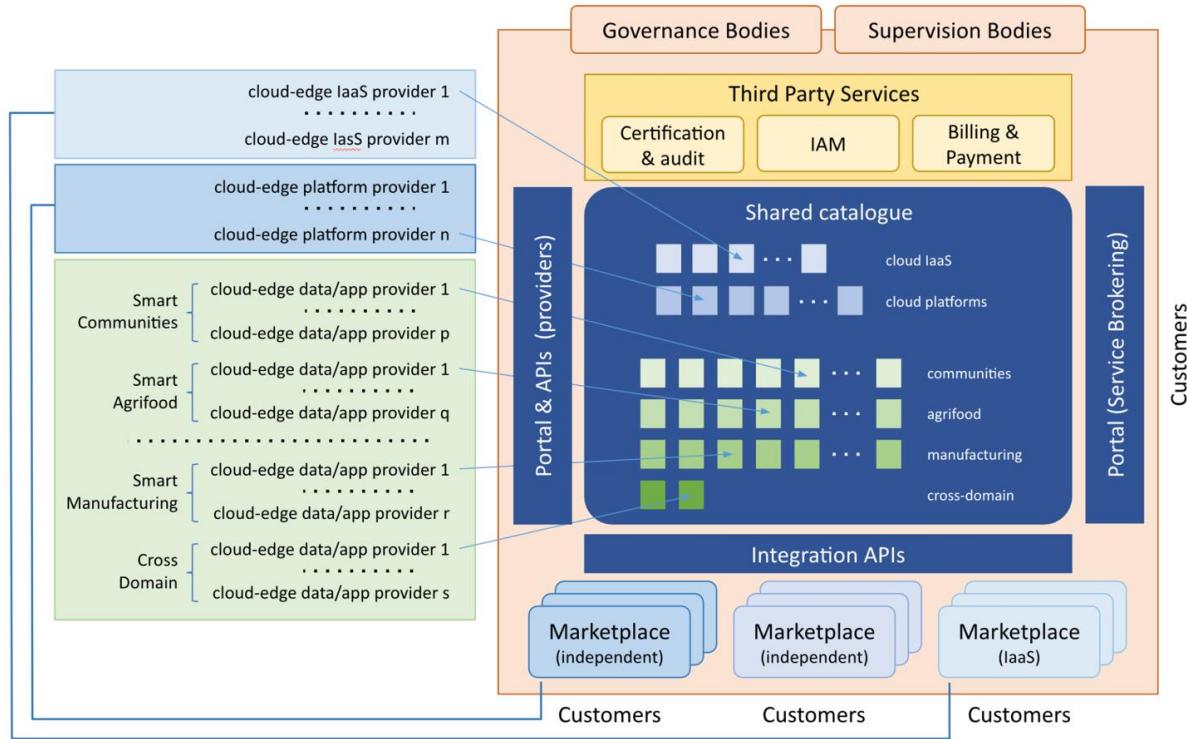


Figure 1- High-level vision of DOME architecture, operating model and roles

2.1.1 Cloud and edge service providers

Cloud and edge service providers (IaaS, platform and app/data service providers) are organisations (public institutions or private companies) that offer service products that can be consumed by customers, such as other organisations or individuals. They access the DOME provider portal where they can register and manage the description of specifications and offerings linked to their products. **Product Specifications** and **Product Offerings** associated with a given service from a service provider are stored in the **Shared Service Catalogue** that is the central part of DOME. A Product comprises a number of Services and supporting Resources (e.g., an Air Quality monitoring product for a given City may consist of an application offered as a Service from the Cloud and a number of computing resources on the Cloud plus a number of IoT devices for monitoring air quality deployed in the field).

The description of each Product Specification and Product Offering will be provided in a standard-based format prescribed in DOME. The description of a Product Specification will comprise information like the unique identifier of the product, its name, version, associated documentation, description of software services implementing the product functionality, description of resources required for execution of such software services (e.g., computing capacity, including disk storage, to be provisioned for serving each customer, or devices to be deployed on the edge), status within the lifecycle of the product (under testing, validated, active, obsolete, retired, ...), etc. On the other hand, the description of a Product Offering will comprise a unique identifier, a reference to the specification of the product being offered, lifecycle status, terms and conditions associated to its use, pricing model, associated agreements (e.g., list of Service Level Agreements that users can choose from), target market segment, kind of marketplaces through which the product can be offered, etc. Both, Product Specifications and Product Offering descriptions, will comprise a number of labels issued by

certification agencies in connection with the service offered that certifies compliance with defined EU regulations or rules established by supervision authorities (e.g., GDPR regulations, established regulations for specific sectors like health, energy, finance, regulations for cloud services to be established in the EU Cloud Rulebook, ...), relevant standards (e.g., standards for interoperability) or best practices (e.g., Open Source Security Foundation Best Practices).

Cloud and edge data/app service providers could receive **Product Orders** from end customers directly from the DOME order management system or from marketplaces federated in DOME which have incorporated the data/app services as part of their catalogue (see description of the role of Federated Marketplaces below). Similarly, providers may receive payments through third-party payment service providers that have integrated their services directly with DOME, but they can receive such payments also from the payment services implemented by the federated marketplaces through which service orders were issued.

Through specific pages for providers of the DOME portal, cloud and edge IaaS, Platform and data/app service providers can also monitor the evolution of instances of their services for particular end users and generate different kinds of reports. In order to be able to access these specific pages for providers under the DOME portal, each cloud and edge IaaS, Platform and data/app service provider has to be registered in the eIDAS service. See objective #2.

2.1.2 Federated Marketplaces

As illustrated in figure 1, different kind of marketplaces can be federated to DOME:

- **Marketplace connected to an IaaS provider**, which comprises a catalogue of cloud and edge data/app services which customers can pick and then easily deploy on top of the computing infrastructure supported by the given IaaS provider
- **Marketplace connected to a Platform provider** which comprises a catalogue of cloud and edge data/app services which customers can pick and easily activate, and will run integrated with the rest of data/app services already running for the customer on top of the provided Platform.
- **Independent Marketplace**, which comprises a catalogue of cloud and edge data/app services which are not tied to any particular IaaS or Platform provider

Examples of Marketplace connected to Platform providers would be marketplaces connected to specific application domains, like Smart Cities or a Smart Farming, or marketplaces connected to specific technology platforms, like a Spark-based platform for development of AI apps, or a Grafana-based platform for development of dashboard apps. In the case of a marketplace connected to a specific Smart City platform, the catalogue may comprise apps for Smart Parking, Smart Air Monitoring or Smart Waste Management, for example. Note that each data/app service may be hosted on a different IaaS cloud or servers and it does not need to be the same where the Platform is hosted. In the case of a Smart Farming Platform, the catalogue may comprise apps for Smart Field Watering, Smart Pesticide Spreading or Smart Silos Management. Similarly, a marketplace connected to a Spark-based platform may comprise applications for predictive maintenance of vehicles, or Weather predictions. Some of the data/app services can be provided by the Platform provider (e.g., Integrated Command and Control system in connection with Smart City Platforms, or Smart Farm Management Information System in connection with Smart Farming Platforms). Some of them may be already active by default for all customers, otherwise may require acquisition through the marketplace.

Note that a given cloud/edge service may be visible in multiple marketplaces. On the other



hand, a given marketplace may only comprise a subset of the cloud and edge services listed in the DOME shared catalogue (e.g., the Marketplace connected to a concrete Smart City platform will only include data/app services relevant for cities).

Note that cloud and edge data/app services will always be visible and could be procured through DOME. However, federated Marketplaces will typically bring a personalised user experience to their target customers, and also a different implementation of their own rating, billing and payment processes, even though they may rely on payment and billing services offered by third parties through DOME.

2.1.3 Customers

European public and private customers looking for trusted cloud and edge services will interact with DOME following one of the two following paths:

- In a very first step, accessing the DOME portal and leveraging service brokering functions of the DOME technical infrastructure to discover IaaS or platform providers which, together with their associated marketplaces can bring to them the best personalised experience. Afterwards, interacting directly through the marketplace associated with the IaaS and platform of their choice, picking the concrete cloud and edge data/app services offerings that are published through the marketplace catalogue which therefore can be seamlessly integrated with their selected IaaS/Platform to support processes of their organisation.
- Accessing the DOME portal to find cloud and edge services directly, placing and managing orders of selected services via DOME, and conducting payments via payment systems which are supported directly by the service provider or are offered by third parties connected to DOME and accepted by the selected service provider.

While the second path will be feasible, it is envisaged that the first path will be more optimal, since the consumer will benefit from a more rich and comprehensive service and user experience that IaaS/Platform providers can offer (see more in connection to Objective #4).

2.1.4 Operators of the DOME technical infrastructure

During the execution of the project, a number of companies of the consortium will act as operators of the DOME technical infrastructure, ensuring the proper functioning of DOME, including security aspects. Along the duration of the project a revenue-based business model will be defined to ensure long-term sustainability of DOME evolution and operations. Such business model may imply creation of an independent company, institution or association by the operators (see description of Objective #9).

2.1.5 Third parties integrating and offering complementary services

DOME will provide means for integration of Third-party services, namely:

- Services from certification and audit agencies which will help to validate the reliability, security, and sovereignty of certain cloud services by checking/verifying their compliance with predetermined market-wide certifications (see more in description of



Objective #6).

- IAM service providers offering services aligned with open standards for IAM adopted in DOME (see more in description of Objective #2), bringing participants the ability to securely manage identities and access to specific cloud and edge data/app services.
- Billing and Payment service providers working as gateways that rely on transaction logs registered in the federated blockchain network infrastructure underlying DOME to provide secure, transparent and trustful billing to consumers and payment to providers.

For all these three kinds of third-party services, the marketplace represents a new source of revenue, as it gives them access to a new market (the cloud and edge service providers and the customers). On the other hand, they represent potential sources of revenue for securing the sustainability of DOME.

Integration of third party IAM providers will not be required for DOME to function since DOME will already provide an implementation of the IAM framework since its very first version, with open source components that the different parties (providers, consumers, marketplaces) may integrate in their runtime environments. However, integration with third party IAM service providers that implement the same standards will be supported. On the other hand, Billing and Payment Service providers will not be required for procuring data/app services through federated marketplaces since they will typically integrate their own billing and payment systems. Integration with third party Billing and Payment service providers will be useful when data/app service providers opt for activating procurement directly through DOME. For these reasons, the ability for third party IAM, Billing and Payment service providers to integrate their services with DOME will be incorporated as one of its functions.

2.1.6 Members of governance and supervisory bodies

Last but not least, DOME will define suitable governance and supervisory bodies that will oversee development of the ecosystem around DOME ensuring fulfilment of its objectives (see more details in connection to objective #10). These bodies will incorporate not only initial consortium members and representatives of relevant stakeholders including the EC and representatives of member states but will be open to the addition of new members (e.g., providers of marketplaces federated with DOME after the projects starts).

2.2 Overall technical approach

2.2.1 Basic Information model

DOME relies on a subset of TM Forum Open API recommendations with regards to the definition of its underlying information model as well as the implementation of APIs that support marketplace federation and the generation of logs during the lifecycle of cloud and edge services and service offerings.

Following TM Forum recommendations, DOME supports the concept of **Product (Offering) Catalog** which is a collection of **Product Offerings** published by Providers through a set of specific Distribution Channels (e.g., federated marketplaces or DOME itself) and targeted to



Market Segments¹. Here, we use the term **Product** to align with the terminology used in TM Forum recommendations which refers to the particular instantiation for a Customer of a set of related Services (e.g., a set of software services offered through a RESTful API on a particular endpoint) and required Resources (e.g., the concrete devices that had to be deployed on the consumer premises or storage and VM capacity that needed to be allocated in the cloud to support execution of software services associated to the Product). A **Product** complies with a given **Product Specification** which in turn refers to the **Service Specifications** and **Resource Specifications** that Services and Resources realising the Product have to comply with.

Product instances as well as the associated services and resources instances are registered in the **Product, Service and Resource Inventory** when they are instantiated.

When a Customer wishes to get a Product instance materialised, it has to issue a **Product Order**. Only when that Product Order has been successfully completed, a Product instance gets materialised for the Customer. That Product instance is bound to a contract between the Customer and the Provider of the associated Product Offering.

A Product Offering comprises elements such as a reference to the corresponding Product Specification, the agreements that govern usage of the served product, a productOfferingPrice, the marketSegment it is targeted to, channels through which it can be offered, and other aspects which characterise the products created when the Product Offering is procured. Note that there may be one or more Product Offerings around the same Product Specification (e.g., associated with different prices or targeted to different market segments).

Each time a Resource or Service associated with a Product is used, a **Usage Log** is created, which typically is used to calculate how much can be charged to Customers and paid to Providers.

The described information model is represented in the following figure:

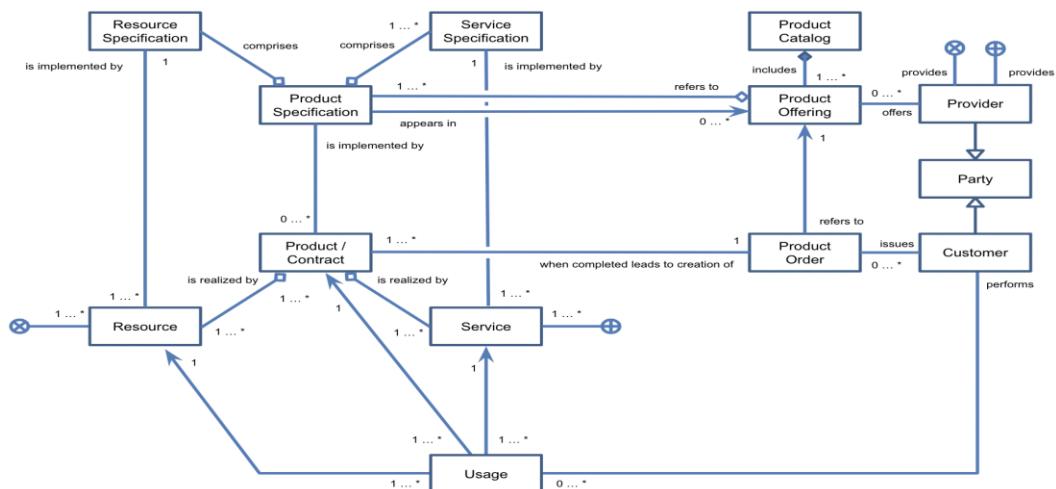


Figure 2 - TM Forum information model applied in DOME

¹ Note that in TM Forum recommendations the term “Product” is used instead of “Service”. This is because marketplaces relying on the specifications can be defined for different kind of products, including cloud and edge services but also any kind of physical or digital products/assets.

2.2.2 DOME Shared Catalogue and Transactions Ledger (Distributed Persistence Layer)

At the heart of the technical architecture of DOME is the **DOME Distributed Persistence Layer** which manages storage of, and access to, information associated with:

- the Shared Catalogue of Product Specifications (including the specifications of associated services and supporting resources) and Product Offerings defined by cloud and edge service providers
- Product Orders and Product instances along their lifecycle, as well as information about actual Usage of Products

The DOME Distributed Persistence Layer will be implemented on top of a number of interconnected national blockchains (starting with Alastria and HashNet) compatible with the European Blockchain Service Infrastructure (EBSI) when not directly EBSI. As illustrated in Figure 3, each cloud and edge service provider, federated marketplace, and the DOME Portal backend itself implements an access node to the DOME Distributed Persistence Layer that supports the standard TM Forum APIs defined for the implementation of Marketplace functions.

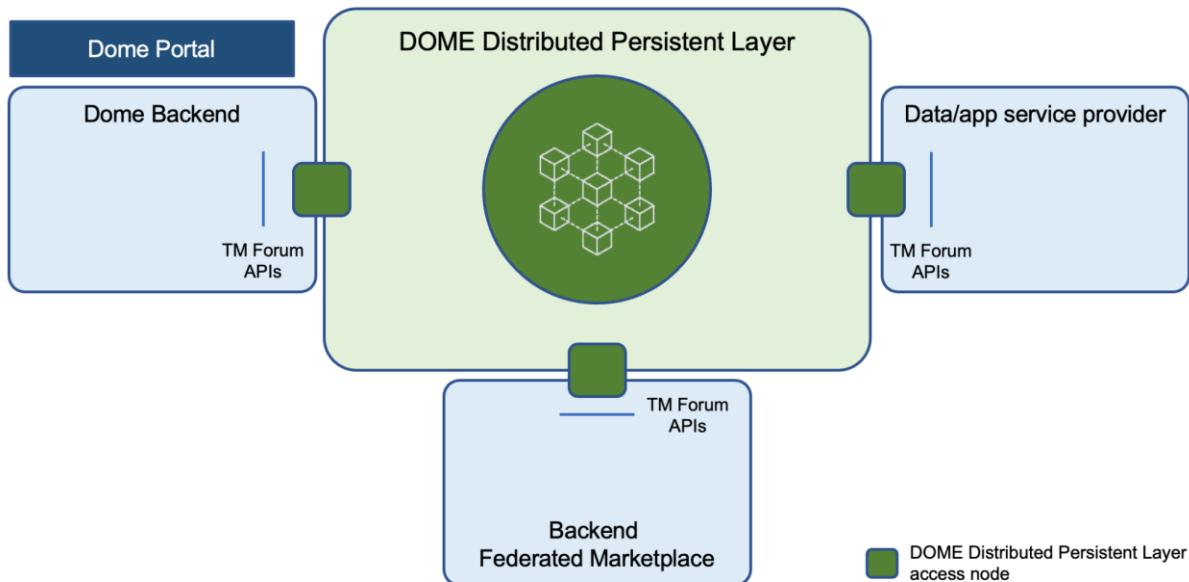


Figure 3 - High-level architecture of the DOME Distributed Persistence Layer

When a Product Offering is created through the DOME Portal, for example, information about it has to be stored in the DOME Distributed Persistence Layer. This is achieved by invoking the specific operation for creating a Product Offering entity of the TM Forum Catalog Management API (TMF620 recommendation) that the Distributed Persistence Layer access node implemented in the DOME Portal backend supports. Part of this information is stored in the blockchain and, consequently, becomes replicated in all other nodes connected to the

DOME Distributed Persistence Layer while the rest of the information will be stored “off-chain” within the access node, which will typically also store a local copy of the information stored in the blockchain to support local queries in a more efficient manner. During the first phases of the DOME project, a decision will be made about what part of the information will be stored in the blockchain and what part of information will be stored only “off-chain”. In any case, any access node will be able to access information stored “off-chain” based on information stored in the blockchain, provided it owns the necessary credentials that grant them access to the nodes where such “off-chain” data is stored.

Aligned with Gaia-X specifications, the description of Product Specifications and Product Offerings will be represented in the form of Verifiable Credentials/Presentations (VC/VP) compliant with W3C standard specifications², some of which will take the form of labelled certifications (verifiable credentials issued by certification and audit agencies). These VC/VPs are stored in the DOME Distributed Persistence Layer.

The DOME Distributed Persistence Layer brings transparency and trust to all participants since all transactions linked to the creation of Product Specifications, Product Offerings, Product Orders and Product Instances as well as their evolution over time or the generation of Usage Logs will be stored in a blockchain. This allows, for example, cloud data/app service providers to audit when their services have been procured and through which marketplace (any of the federated ones or directly DOME). Similarly, it will allow a given marketplace provider to audit when a given data/app service that was procured through its marketplace has been used. Last but not least, Usage Logs can be used by third party Charging/Billing/Payment gateways integrated with DOME which may be offered to cloud and edge service providers which do not want to implement a charging/billing/payment system on their own. They can also be used to generate verifiable credentials regarding operations of a service provider which can later be used as “passport” in front of investors or funding agencies.

2.3 Journey of DOME users

2.3.1 Service providers journey

Figure 4 describes the journey that cloud and edge service providers will go through when interacting with DOME.



Figure 4 - Service providers journey in DOME

² W3C Verifiable Credentials Data Model v1.1, W3C Decentralized Identifiers (DIDs) v1.0



Next, we explain each of the stages³, providing some details about what goes on within each stage from a technical perspective:

- Stage 1 - Subscribe

The ‘subscription’ stage comprises all the steps followed by any given cloud and edge service provider since it joins DOME until it publishes its Product Offerings. This consists of three steps that the provider performs via DOME (either through APIs or the Portal) – 1) registration as a cloud/edge service (product) provider, 2) registration of the specifications of products it offers (defined as combination of services and associated resources) as well as definition of the basic characteristics of product offerings around registered product specifications like market segments the offering is targeted to (useful later on to fine tune discovery services), sales channels through which the offering will be visible (e.g., type of federated marketplaces in addition to DOME), or terms and conditions, including information about the different pricing models supported, and 3) verification of the compliance with DOME’s basic standards and criteria.

All these steps will imply registration and management of information linked to entities described in the information model previously described in Figure 2 using TM Forum APIs that the DOME Distributed Persistence Layer supports. As an example, registration of a given cloud/edge service provider would mean creation of a Party playing the role of Provider using the TM Forum Party Management API (TMF632 recommendation). Similarly, registration of Product Specifications and Product Offerings will be performed using the TM Forum Product Catalog Management API (TMF620 recommendation) which in turn will rely on the TM Forum Service Catalog Management API (TMF633 recommendation) and the TM Forum Resource Catalog Management API (TMF634 recommendation) since products are made out of the combination of services and supporting resources. Cloud and edge service providers can perform these operations programmatically using the TM Forum APIs that their access nodes to the DOME Distributed Persistence Layer support or via the DOME Portal (whose backend, on the other hand, uses TM Forum APIs supported by the DOME Distributed Persistence Layer). Compliance verification of a given Product Specification or Product Offering will imply the transition of their status (one of the attributes these kinds of entities export) into “active” status.

An IaaS or Platform service provider that has implemented a marketplace connected to its services also relies on the TM Forum Party Management, Product Catalog Management, Service Catalog Management and Resource Catalog Management APIs (TMF632, TMF620, TMF633 and TMF634 recommendations) that their access nodes to the DOME Persistent Layer offer in order to register data/app service providers, as well as corresponding Product Specifications and Product Offerings through their marketplaces instead of directly through DOME. This is why we say that these marketplaces connected to IaaS or Platform service providers are federated to DOME: no matter how a data/app service provider registers, directly or through a federated marketplace, its Product Specifications or Product Offerings will end up

³ The following discussion is inspired by the study available at <https://digital-strategy.ec.europa.eu/en/library/building-european-cloud-marketplace-conceptualisation-study>



registered in the DOME Share Catalogue (part of the DOME Distributed Persistence Layer).

As mentioned before, part of the information describing Parties (cloud and edge service providers), Products and Product Offerings will take the form of Verifiable Credentials (VCs). Verification of some of these VCs will be required for a Party, Product or Product offering to be registered successfully in the DOME Persistent Layer, and therefore become visible through the DOME portal or any of the federated marketplace portals. This way, DOME provides guarantees to end customers that Parties, Products and Product Offerings that are offered through the DOME portal or the portal of any of its federated marketplaces are curated and will meet certain regulations and characteristics. Note that verification of a given credential will imply verification that the issuer of the credential is in the list of Trusted Issuers of Verifiable Credentials that DOME will rely on (and will be stored in the blockchain).

Cloud and edge service providers get notified when information relevant to them is stored in the DOME Persistent Layer. Such notifications are received through their access nodes to the DOME Persistence Layer. Thus, for example, when a specific data/app service provider registers a given product offering (associated with a given product specification defined as combination of services and resources) in DOME, it will be offered the possibility of registering the product offering just in DOME or in DOME as well as any of the marketplaces connected to IaaS or Platform service providers federated with DOME. In the latter case, these IaaS or Platform service providers will receive a notification through the access node to the DOME Persistent Layer they implement. This way, a provider has only to register a data/app service once in DOME and get visible through the catalogue of all federated marketplaces it allows to work as sales channels. Note that additional compliance verification may be performed at the level of each of the federated marketplaces. For example, support of a NGSI-LD interface by the data/app service being registered may be verified by marketplaces associated with Platform services that are based on FIWARE.

Figure 5 illustrates interactions that take place during the Subscribe stage.

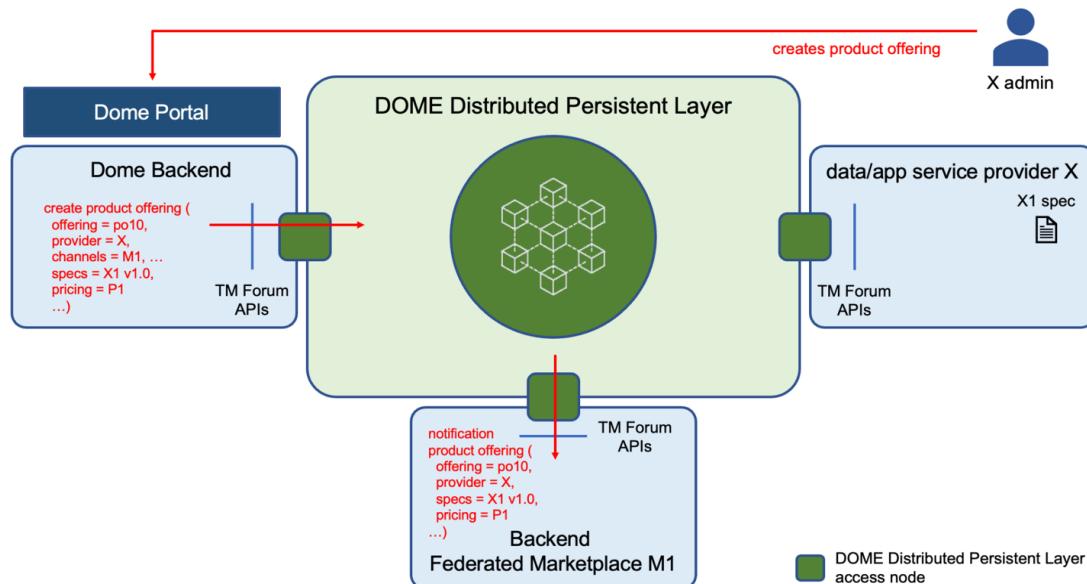


Figure 5 - interactions among components during subscribe stage



- Stage 2 - Reference

Once a Product Offering becomes “active” it becomes visible to other users of DOME (typically end customers as well as IaaS and Platform service providers in the case of data/app services, since they may be interested to incorporate those data/app services in their respective catalogues). The provider of the Product Offering may establish visibility rules that determine who can get access to the offering.

Cloud and edge service providers may refer to web pages of the DOME portal describing their product offerings once incorporated in the DOME Catalog. This way being able to promote them in front of potential customers.

A Cloud and edge service provider is able to update characteristics of its Product Offerings as well as corresponding Product Specifications (or specifications of associated services and resources). Those updates can be formulated through TM Forum APIs supported by the access nodes to the DOME Distributed Persistence Layer or via the DOME portal. These updates will not only get registered in the DOME Product Catalog (becoming then visible through the DOME Portal to other direct users) but will be propagated to federated marketplaces in which the given Product Offerings / Specifications that got updated were also registered. This propagation will take place through notifications that federated marketplaces will receive through the access nodes to the DOME Distributed Persistence Layer they implement.

Through search and browsing capabilities that the DOME Portal will implement, customers will be able to easily find the specific product they are looking for. In its most basic format, the DOME portal will allow customers to launch product offerings and product specifications searches, leveraging category filters and tagging functions, some tags connected to Verifiable Credentials (VCs) describing them. These functions will also be accessible via API enabling integration of more sophisticated customer applications.

Beyond basic search functions, DOME will implement more advanced features with the goal of connecting consumers with relevant services as quickly as possible. Among the other features, it will be possible to implement a search algorithm which would match customer search queries with keywords from relevant product listings. Even more advanced search functions may leverage additional information (such as product ratings or click-through rates) to prioritise/rank the results of search queries and improve the customer experience. Finally, search algorithms could also be specific to the customer’s sector to provide results that take into account the customer’s particularities.

Figure 6 illustrates interactions that take place during the Reference stage.



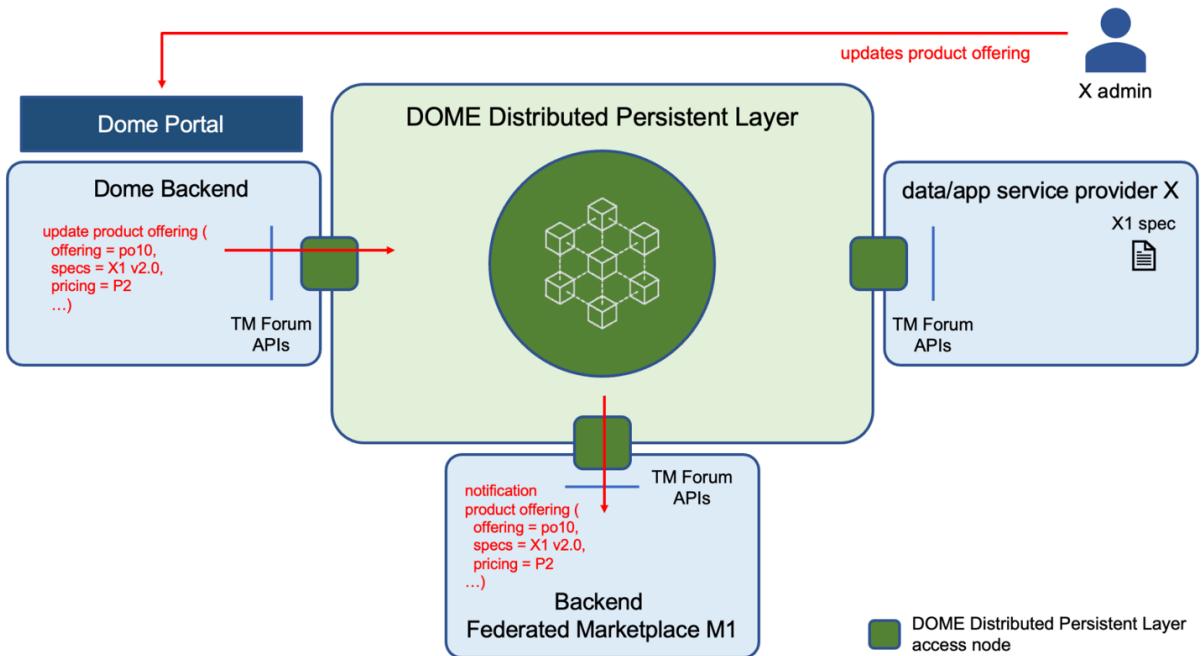


Figure 6 - interactions among components during reference stage

- Stage 3 - Sell

There are two ways in which a given Product offered by a cloud/edge service provider can be procured: either directly through DOME or through marketplaces associated with IaaS or Platform service providers where the corresponding Product Offering has also been registered. When a given customer discovers a cloud/edge service Product Offering it is interested in, both possibilities are offered.

In the first case, procurement may be performed either via the DOME Portal or programmatically. In both cases, the creation of a Product Order will be ultimately requested using the TM Forum Product Ordering Management API (TMF622 recommendation) that the Distributed Persistence Layer access node implemented in the DOME Portal backend supports.

In the second case, typically associated with procurement of data/app services, the customer will be redirected to the marketplace of its choice, through which the procurement process will be handled. At a given moment, the creation of a Product Order will be performed via invocation of the TM Forum Product Ordering API supported by the DOME Distributed Persistence Layer access node linked to the selected marketplace. Note that this Product Order will also become visible not only in the federated marketplace but also at the DOME Portal.

In any of the two cases, a Product Order is created within the DOME Distributed Persistence Layer and the given cloud/edge provider will receive a notification about creation of the Product Order it should handle. This notification will be received through their corresponding DOME Distributed Persistence Layer access node.

Note that many customers will end up consuming services through the portals of federated marketplaces the DOME portal will guide them to. This is because these

portals are expected to provide a better tailored user experience (UX). However, the federation of marketplaces with DOME will mean that all relevant transactions will be registered in the DOME Distributed Persistence Layer and therefore become visible at the DOME Portal, this way ensuring transparency and giving higher trust to both customers and data/app service providers.

Figure 7 illustrates interactions that take place during Product Ordering.

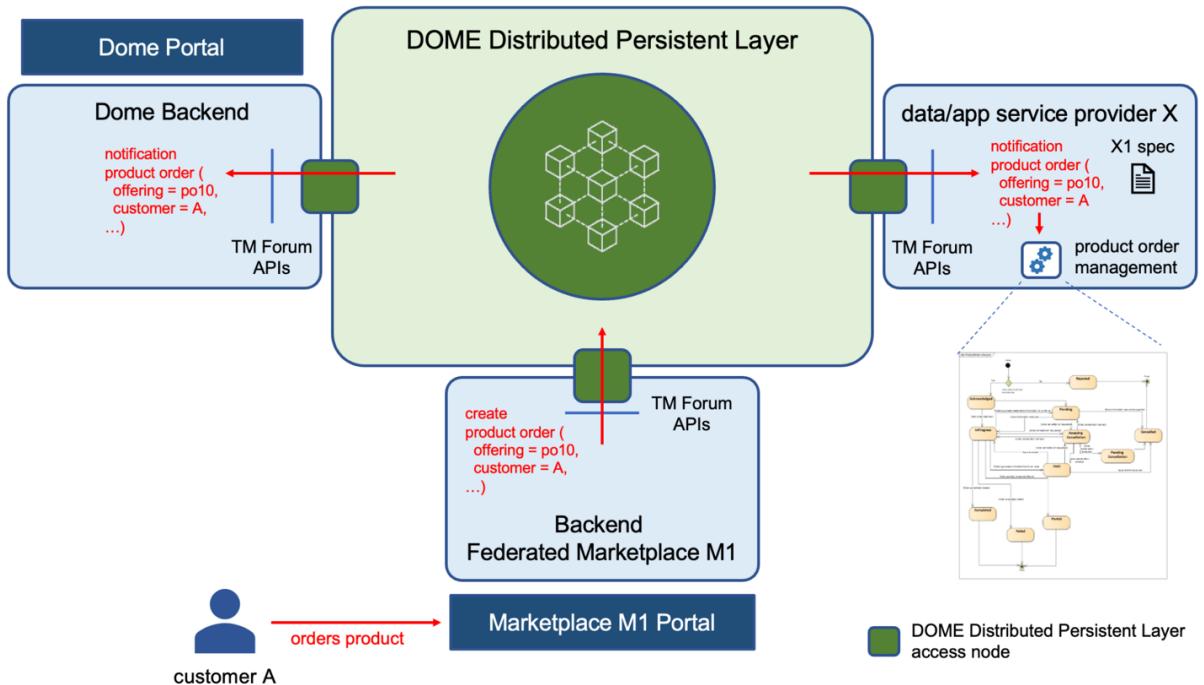


Figure 7 - interactions among components during Product Ordering

Figure 8 illustrates the different states a Product Order will go through since it is issued by a given customer and it gets completed. Such states will be reflected as values of the attribute “state” that any Product Order will support. The defined lifecycle complies with TM Forum specifications but will be revised based on feedback from first deployment and pilots of DOME.

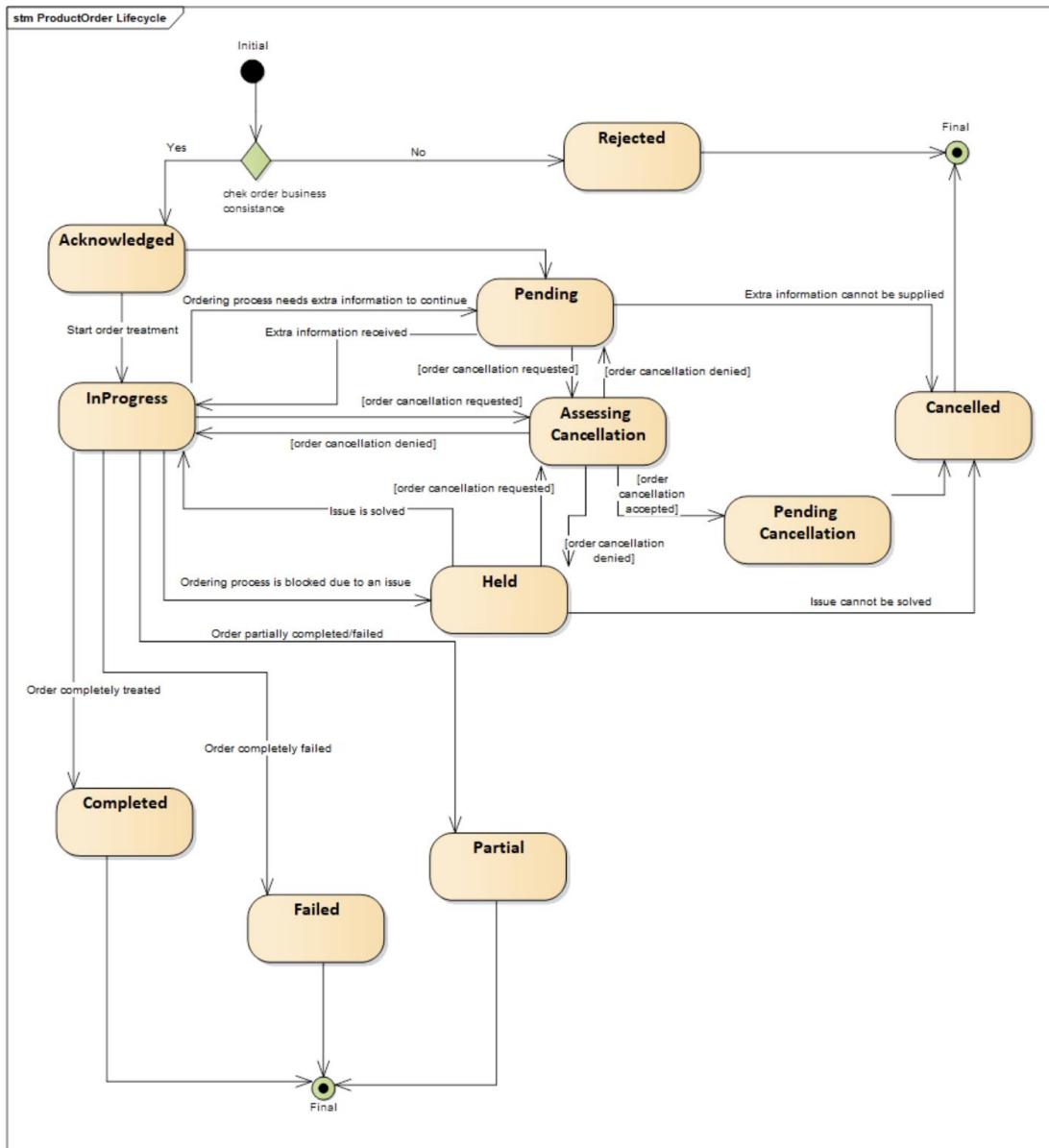


Figure 8 - Lifecycle of Product Orders

Once a Product Order is completed, a contract between the customer and the service provider is established so that terms and conditions defined in the Product Offering start to apply. As a result, the customer becomes a Trusted Issuer of Verifiable Credentials relevant to the product business logic (see section on “Trust Anchor and Decentralized Identity and Access Management (IAM) Framework” below). The service provider will then create a Product instance using the TM Forum Product Inventory Management API (TMF637 recommendation) that its access node to the DOME Distributed Persistence Layer supports. As a result, the Product instance will become visible to the customer, either through the DOME portal or the federated marketplace through which the originating Product Order was issued.

Figure 9 illustrates the different states a Product instance will go through since it is created, right after the originating Product Order was completed, until it is terminated. Such states will be reflected as values of the attribute “state” that any Product instance will support. The defined lifecycle complies with TM Forum specifications but will be revised based on feedback from first deployment and pilots of DOME.

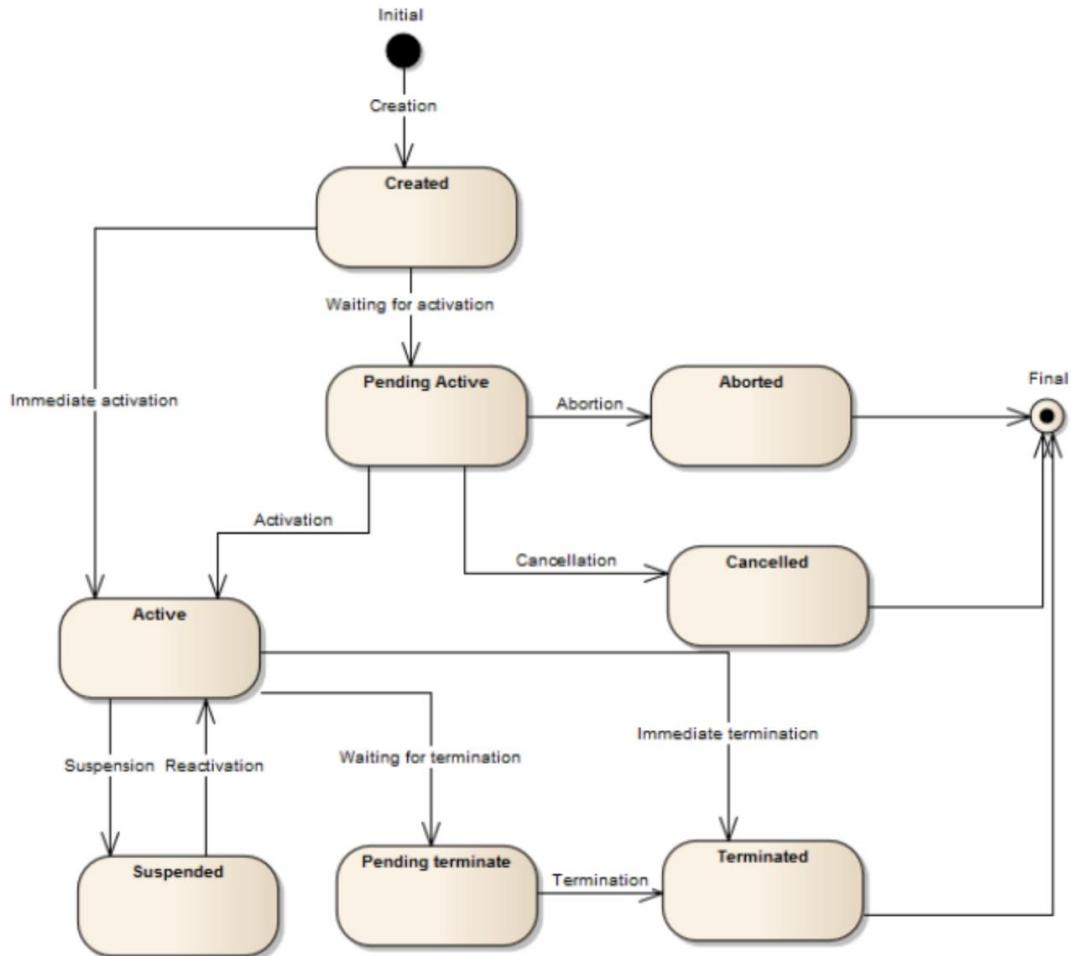


Figure 9 - Lifecycle of a Product (instance)

Note that the creation of a Product instance does not necessarily mean that its component services and required resources get automatically provisioned and activated. There may exist a period from the time at which a Product is created until it actually can be used by the customer that ordered it. This is for example the case in connection to products which require deployment of resources in the field (e.g., an app for air quality monitoring which requires deployment of several IoT devices in the field). It is also the case when manual configuration and/or integration testing with products from third parties is required. Once everything is ready for actual usage, the state of the Product becomes “Active”. This will be the point at which access to the service will be permitted, or logs for the initial charging will be generated in connection to one-payment or subscription fee pricing models. It will also be the point at which Usage

logs will start to be generated, bringing the basis for the monitoring of services as well as the support to pay-per-use pricing models.

Cloud and edge service providers registered in DOME will commit to register Usage logs in the DOME Distributed Persistence Layer, using the TM Forum Usage Management API (TMF635 recommendation) that its access node to the DOME Distributed Persistence Layer supports. Those Usage logs will ultimately be recorded in the blockchain associated with the DOME Distributed Persistence Layer but multiple logs will be condensed into a block for performance reasons.

Figure 10 illustrates interactions that will take place during the lifecycle of a Product instance, particularly at the time of its creation as the result of completing a Product Order by a particular customer, and its activation for that customer.

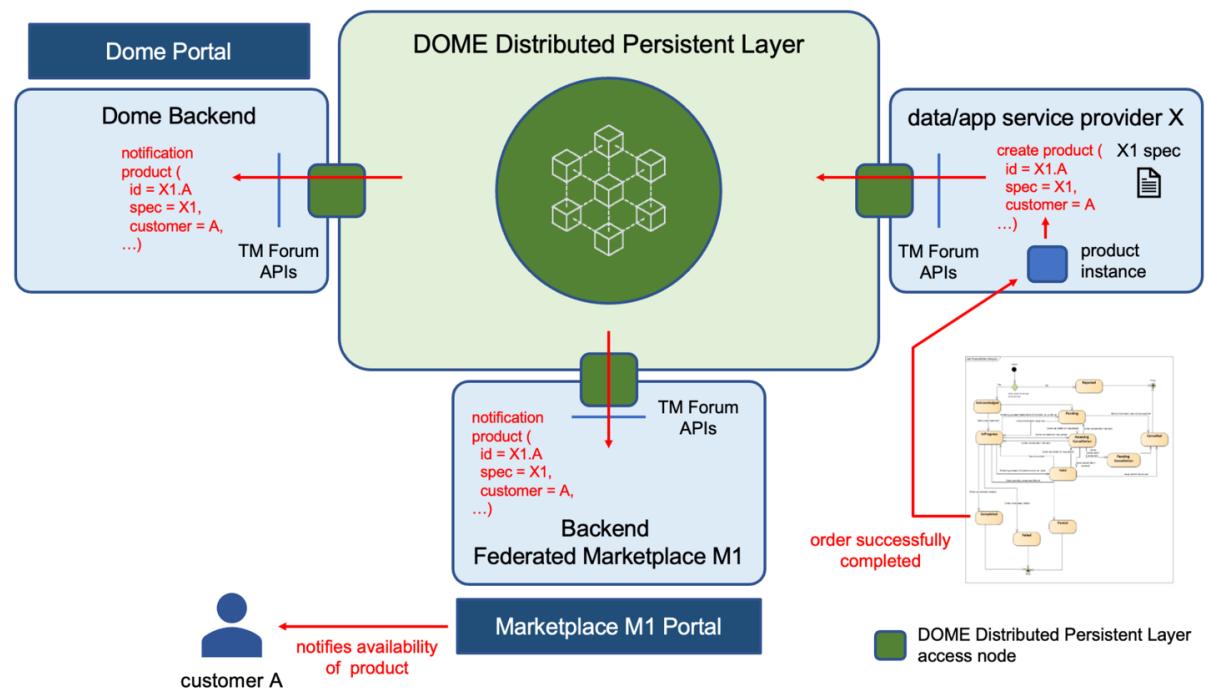


Figure 10 - Interactions during the lifecycle of Product instances

- Stage 4 - Follow

Once providers have sold their service, they will need to be able to monitor consumption, provide after-sale support and leverage the experience to innovate and continuously strengthen their service offering.

Reporting and analytics features will be offered via the DOME Portal that providers and customers can consume. Notably, the marketplace can provide users the option to personalise their reports or to export data to outside platforms via connectors and APIs. Since relevant information for fueling these reporting and analytic tools is accessible through the DOME Distributed Persistence Layer, more advanced versions can be offered as Third-Party services that get access to the APIs that the DOME Distributed Persistence Layer offers.

Note that customers will mostly end up consuming services through the portals of federated marketplaces the DOME portal will guide them to. These portals are expected to incorporate their own reporting and analytic functions meeting the needs of customers (particularly to support the different stages of their journey).

Some of the cloud or edge data services registered in DOME may bring access to static data or near real-time data resources available through RESTful APIs (e.g., IoT data). DOME will integrate data publication functions enabling the exposure of such data resources in compliance with DCAT specifications defined by W3C and DCAT-AP recommendation by the EC. This way, data resources linked to data services offered through DOME can be harvested through external Data Publication platforms (e.g., the European Data Portal).

2.3.2 Customers journey

Figure 11 describes the journey that consumers of cloud and edge services offered through DOME will go through when interacting with DOME.



Figure 11 - Consumers journey in DOME⁴

Those customers who approach the DOME ecosystem for the first time or wish to check other marketplaces different than the one they are already using, will connect to the DOME portal searching for offerings. They may end selecting and contracting individual data/app cloud or edge services directly through DOME which may require use of third-party payment services integrated with DOME. However, in other cases they will look for the marketplaces, connected or not to an IaaS, Platform provider or Individual Marketplace Provider that may better solve their overall needs. When a customer selects a given marketplace then they will further interact with its corresponding portal, which will typically mean they will enjoy a more personalised user experience through that portal interface, including the payment of services. Note that, despite further interactions will then be bilateral with the marketplace, DOME will bring trust to the relationships established between customers and app/data cloud and edge services, because both can audit transactions as they are logged in the DOME Distributed Persistence Layer. Satisfied customers will become the best ambassador of DOME and federated marketplaces based on a satisfactory experience.

⁴ figure taken from the conceptualization study “Building a European Cloud Marketplace”, entrusted by the European Commission to CapGemini Invent

3 DOME Trust and IAM Framework

DOME brings a decentralised Trust and Identity and Access Management (IAM) framework to enable the trusted operation with the system without requiring a central entity intermediating in all interactions.

The Trust framework ensures that the information published on DOME is trustful. It defines and enforces a set of rules that actors in the DOME ecosystem (data/app service providers, federated marketplace providers, end customers) agree to follow. By following them, all organisations can use their digital identities and characterise services in a consistent and trustful manner. This lowers the barriers for organisations to complete transactions or share information with other organisations.

The IAM framework, on the other hand, enables actors in the DOME ecosystem to authenticate into DOME services (i.e., the DOME Portal, federated marketplace portals or TM Forum APIs implemented on top of the DOME Persistence Layer) and help to manage proper access to those services based on their profiles. This IAM framework relies on Verifiable Credentials/Verifiable Presentations and leverages the Trust framework to provide an efficient, scalable, and decentralised IAM that participants can use. The Trust and IAM framework implemented in DOME is not only for managing authentication and authorization in the interaction with the DOME services, but it is also available for data/app services which may use it for managing the interactions between them and their users.

3.1 Trust Framework

The Trust framework addresses the following issues:

- **ID Binding:** How to verify that a given identifier corresponds to a valid legal identity of an entity in the real world?
- **Proof of participation:** How to verify that the entity is trusted because it is a subscribed participant in a given ecosystem (e.g., the DOME ecosystem)?
- **Proof of Issuing Authority:** How to check that the credentials presented by a participant have been issued by another entity that can be considered a Trusted Issuer of that type of credentials? This enables the verifier to put the right level of trust in the facts attested by the Verifiable Credentials presented by a participant.

These concepts are elaborated in the next sections.

3.1.1 ID Binding

At the root of any trust framework there is the requirement to verify the identity of an entity in the real world and the assignment of some identifier that can be used later in representation



of the real entity in the online processes. This association between an identifier (including some metadata) and the real identity of an entity is what we call **ID Binding**.

Please note that, at this level, ID Binding states only who the entity is in the real world, not any additional properties that may be interesting for other purposes. For example, ID Binding establishes that the entity is a business operating in the EU, but it does not say what products it sells or the characteristics of the product, or the markets in which it operates, or in which data spaces it participates.

Many ecosystems assign a proprietary identifier to entities when they are onboarded in the ecosystem, creating silos of identifiers, and making very difficult the interoperability across ecosystems.

In DOME we are limiting the ecosystem to those entities that can operate in the EU. Or more precisely, those entities that can engage in electronic transactions in the internal market of the EU by using advanced or qualified electronic signatures/seals, as defined in the eIDAS regulation. In practice, this means that in order to participate in DOME the entities should be able to obtain a digital certificate from any Trust Service Provider (TSP) listed in a Member State national Trusted List (eIDAS articles 22.1 and 22.4).

In addition, in the first phase of implementation of DOME we require that the entities that participate directly in the ecosystem be legal entities, so the certificates that enable onboarding in DOME are those issued either to legal persons or to natural persons who are legal representatives of legal persons. Under eIDAS, when a transaction requires a qualified electronic seal from a legal person, a qualified electronic signature from the authorised representative of the legal person is equally acceptable.

In DOME we rely on identifiers already used in those digital certificates issued by a TSP. The combination of digital certificates issued by TSPs and Verifiable Credentials contributes to the legal validity and interoperability of the cross-border data-related transactions in the European Union facilitating the cross-border validation of eSignatures, eSeals, and more. Essentially, Verifiable Credentials and Presentations (including those listed as characteristics of Product Specifications and Offerings) used in the ecosystem will be signed using digital certificates, replacing the “traditional” [signed PDFs](#) with a more efficient format that is machine-readable and machine-processable, with the same level of legal assurance.

Some of the characteristics and advantages of using this type of ID Binding are described below.

3.1.1.1 Cross-border use of mutually recognised electronic identification means

We propose that during onboarding of a new DOME user, eIDAS advanced or qualified signatures and seals are accepted.

The [Regulation on electronic identification and trust services](#) for electronic transactions in the internal market (**eIDAS Regulation**) states that in order to contribute to their general cross-border use, it should be possible to use trust services as evidence in legal proceedings in all Member States. DOME is fully aligned with the objectives of the eIDAS regulation, specifically Article 17 of the eIDAS Regulation, where it is said that Member States should encourage the private sector to voluntarily use electronic identification means under a notified scheme for identification purposes when needed for online services or electronic transactions. The possibility to use such electronic identification enables the private sector to rely on electronic identification and authentication already largely used in many Member States at least for public



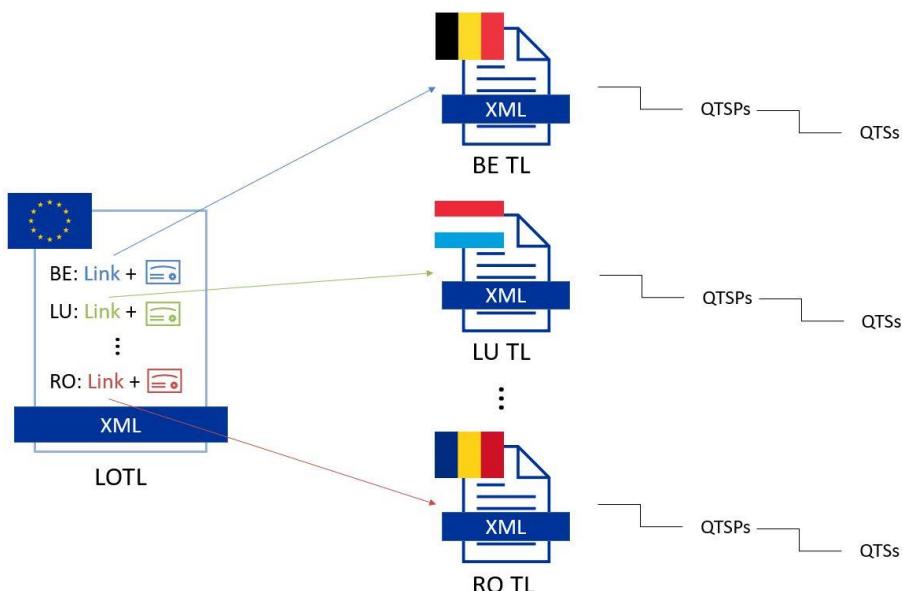
services and to make it easier for businesses and citizens to access their online services across borders.

In this way, interoperability of Verifiable Credentials across the public and the private sector can be achieved in large Digital Ecosystems (e.g., Data Spaces) across the EU.

Article 22 of the [Regulation on electronic identification and trust services](#) for electronic transactions in the internal market (eIDAS Regulation), obliges Member States to establish, maintain and publish trusted lists. These lists should include information related to the qualified trust service providers for which they are responsible, and information related to the qualified trust services provided by them.

In order to contribute to their general cross-border use, it should be possible to use trust services as evidence in legal proceedings in all Member States.

In practice, each Member State publishes its Trusted List, and the Commission publishes the List Of Trusted Lists (LOTL). There are different ways to access the lists, but one which is machine-processable (XML) is located at <https://ec.europa.eu/tools/lotl/eu-lotl.xml> which contains the addresses of each of the Trusted Lists published by each Member State. This is represented in the following figure.



We propose to use an integrated, easy and performant way to perform ID Binding based on the digital certificates provided by the EU TSPs, including transparent access to the consolidated list of TSPs when applications in the Data Space must perform ID Binding (typically during onboarding and verification of signatures of credentials).

3.1.1.2 ID Binding and the Verifiable Credential

In DOME we use a special type of Verifiable Credential intended specifically for authentication and that we will call VerifiableID to distinguish it from the other types of Verifiable Credentials that are also used in DOME for other purposes. When a participant sends online a VerifiableID, in addition to the ID Binding problem described in the previous section, we have to make sure that the credential has been sent by an entity authorised to do so and not by an impostor. This is called ID Binding of the credential. Later in this document we describe a simple approach to perform this using widely used public key cryptography and leveraging on the trust already provided by the digital certificates used in the ID Binding described in the previous section.

Although the process will be described in detail later in the document, we anticipate here the essential characteristics of the credential.

We assume that the issuer of the Credential has an eIDAS certificate (we will use this term for a digital certificate or seal issued by a TSP in the EU Trusted List), and that the issuer is a participant in the relevant ecosystem, e.g., DOME ecosystem or Data Space considered (the concept of participation is elaborated in the next section).

The credential will be signed with the eIDAS certificate, using the [Digital Signature Service \(DSS\)](#) which is a Building Block endorsed by the European Commission to ensure that the digital services are interoperable and EU-compliant. Additionally, DOME will use the standard protocols for remote signature creation defined in [\[ETSI TS 119 432 - Electronic Signatures and Infrastructures \(ESI\): Protocols for remote digital signature creation\]](#), which is based on the specifications from the [Cloud Signature Consortium](#).

In this scenario, the subject of the credential (either a natural or juridical person) wants to receive a credential from the issuer. The credential is an attestation of something that the issuer knows about the subject. That means that there should be a previous close relationship between the issuer and the subject and there is a pre-existing trusted identification mechanism that the issuer uses for everything related to the subject. It could be physical (e.g., a student going to the secretary of the University to request the credential) or electronic (e.g., the student using the “traditional” identification mechanisms for accessing the University online services).

In this context, the approach for issuing verifiable credentials is the following:

1. The subject authenticates to the issuer with whatever mechanism has been used during the previous relationship. In the example of the Diploma, the student uses whatever identification mechanism was provided by the University when the student enrolled in the studies.
2. Once an authenticated session exists, the subject generates a pair of public-private keys and sends the public key to the issuer, keeping the private key private (that's the meaning of its name, obviously). The actual mechanism uses a digital signature of a challenge to ensure that the subject controls the private key associated with the public key sent to the issuer.
3. The issuer creates a credential with the relevant attestations and includes also the public key received from the subject as an additional attestation.
4. The issuer digitally signs the credential with its eIDAS certificate and makes the credential available to the subject. The specific mechanism to “send” the credential to the subject can be diverse. Given that there is a previous relationship, the credential could be sent inside the authenticated session during which the verifiable credential was being requested and it is generated. Alternatively, it could be made available in the “electronic office space” of the subject, or sent encrypted via email, or even printed in physical media and sent via mail, if required. The Verifiable Credential is essentially a file with data in JSON (or JSON-LD) format and digitally signed by the issuer. However, the mechanism for issuance that will be used in DOME is [OpenID for Verifiable Credential Issuance](#), as specified in [The European Digital Identity Wallet Architecture and Reference Framework](#).

With this mechanism, the receiver of the credential can have the same level of trust in the “normal” attested attributes inside the credential and in the public key inside the credential.



The holder of the credential can prove control of that credential by signing challenges from the verifier using the private key associated with the public key that is inside the credential. This way, the credential can be used for both identification and authentication.

3.1.1.3 About identifiers for legal persons

In DOME we use W3C Verifiable Credentials with DIDs as identifiers. A DID is a simple text string consisting of three parts: **1)** the did URI scheme identifier which is the word “did”, **2)** the identifier for the DID method which specifies the mechanism used for resolving a DID, and **3)** the identifier specific to that DID method.

As mentioned before, when using eIDAS digital certificates for identity binding, it does not make sense to “invent” identifiers or to promote the usage of different DID methods that are not well integrated with eIDAS certificates and that generate identifiers which are not in general legally recognised in the EU for economic transactions (e.g., that can be used in electronic invoices across the EU).

In general, an ecosystem may accept one or more DID methods and their associated DID resolution mechanisms (e.g., [did:web](#), [did:peer](#), etc.). For Legal Persons, we propose that the main method used is [did:elsi \(ETSI Legal person Semantic Identifier Method Specification\)](#), which uses as identifiers the same identifiers that are already embedded in the eIDAS certificates that conform to the relevant ETSI standards. The main concepts of the did:elsi method is described below.

ETSI EN 319 412-3 V1.2.1 (2020-07) “**Electronic Signatures and Infrastructures (ESI); Certificate Profiles; Part 3: Certificate profile for certificates issued to legal persons**” states:

“The subject field shall include at least the following attributes as specified in Recommendation ITU-T X.520:

- *countryName*
- *organizationName*
- ***organizationIdentifier*** and
- *commonName*”

And regarding the *organizationIdentifier* attribute it says:

The organizationIdentifier attribute shall contain an identification of the subject organization different from the organization name. Certificates may include one or more semantics identifiers as specified in clause 5 of ETSI EN 319 412-1 [i.4].

And the document referenced, ETSI EN 319 412-1 V1.4.2 (2020-07) “**Electronic Signatures and Infrastructures (ESI); Certificate Profiles; Part 1: Overview and common data structures**” states:

*When the legal person semantics identifier is included, any present **organizationIdentifier** attribute in the subject field shall contain information using the following structure in the presented order:*

- *3 character legal person identity type reference*
- *2 character ISO 3166 [2] country code*
- *hyphen-minus “-” (0x2D (ASCII), U+002D (UTF-8)) and*



- *identifier (according to country and identity type reference)*

The three initial characters shall have one of the following defined values:

- 1) "VAT" for identification based on a national value added tax identification number.
- 2) "NTR" for identification based on an identifier from a national trade register.
- 3) "PSD" for identification based on the national authorization number of a payment service provider under Payments Services Directive (EU) 2015/2366 [i.13]. This shall use the extended structure as defined in ETSI TS 119 495 [3], clause 5.2.1.
- 4) "LEI" for a global Legal Entity Identifier as specified in ISO 17442 [4]. The 2 character ISO 3166 [2] country code shall be set to 'XG'.
- 5) Two characters according to local definition within the specified country and name registration authority, identifying a national scheme that is considered appropriate for national and European level, followed by the character ":" (colon).

Other initial character sequences are reserved for future amendments of the present document. In case "VAT" legal person identity type reference is used in combination with the "EU" transnational country code, the identifier value should comply with Council Directive 2006/112/EC [i.12], article 215.

That means that any eIDAS digital certificate issued by TSPs to legal persons compliant with the ETSI standards including an organizationIdentifier attribute can be used to trivially derive a DID from the ETSI standard identifier by applying the following rule:

did:elsi:organizationIdentifier

Some examples of DIDs are:

- **Gaia-X**: did:elsi:VATBE-0762747721
- **International Data Spaces**: did:elsi:VATDE-325984196
- **Alastria**: did:elsi:VATES-G87936159
- **IN2**: did:elsi:VATES-B60645900
- **Digitel TS**: did:elsi:VATES-B47447560
- **FIWARE Foundation**: did:elsi:VATDE-309937516
- **TNO**: did:elsi:LEIXG-724500AZSGBRY55MNS59

Where:

- "did" is the W3C did uri scheme.
- "elsi" stands for **ETSI Legal Semantic Identifier**, which is the acronym for the name for this type of identifier used in the ETSI documents.
- "organizationIdentifier" is the exact identifier specified in the ETSI standard, and that can evolve with the standard to support any future requirement.

Proving the control of an ELSI DID can be done using the associated digital certificate: including the certificate with any signature can do that. By the way, this means that any existing digital signature of any type of document (not only Verifiable Credentials) is already compliant with this DID method specification, just by making the corresponding translation

In other words: **any legal person can have a standard eIDAS certificate with an automatically associated DID identifier complying with the ELSI did method**



specification. There is no need to invent new identifiers or have a central entity in a Data Space assign identifiers to participants.

3.1.1.4 About identifiers for natural persons

In principle, we could use the same approach for natural persons as for legal persons. The ETSI standards referenced above also cover natural persons, and they define a “Natural Person Semantic Identifier”.

However, legal persons are completely different to natural persons, especially from the point of view of privacy (look at the GDPR to see some differences). It is for those privacy reasons that we propose a different approach to the identifiers of natural persons participating in a sharing ecosystem like a Data Space. In particular, DIDs for natural persons should never be registered in any shared database, including the Verifiable Registry described in the standard W3C Verifiable Credentials ecosystem.

For this reason, DOME uses did:key for natural persons.

3.1.1.5 Disclaimer

The designed and planned architecture, protocol steps, message structures described by the current specification are based on current status of available documents related to eIDAS2 and the EDIW (European Digital Identity Wallet) such as ARF (Architecture and Reference Framework, <https://github.com/eu-digital-identity-wallet/eudi-doc-architecture-and-reference-framework/releases>) and ETSI specifications and reports.

As the development of ARF and ETSI specifications is still in progress the DOME project shall be able to re-factor its solution - on different levels - in case of need.

- The level of communication flows: The DOME project aims at an online communication flow where both Relying Party and User are online, but based on ARF other cases shall be supported in the future as well, even full offline cases where both Relying Party and User are offline entities. The partly offline case is important to support managing e.g. ID documents, national eID cards on the side of the User, therefore it will be important to governments of Member States.
- The level of form factor: The DOME project aims to support a server-side backend wallet with a web application interface, called PWA (Progressive Web Application). Beyond this "web application" form factor also "mobile application" and "secure application on PC" are listed in ARF. Of course, "mobile application" can be also operated with an embedded WebView but for full offline scenarios it should have a standalone AppView as well.
- The level of Verifiable Credentials transmission: The DOME project aims to support OIDC4VCI and OID4VP as it is required by ARF, but these standards contain some alternative configurations and optional attributes and it is not clear whether all these would be supported in the future by the wallet reference implementation or not (e.g. "Pre-Authorized Code Flow" or "Authorization Code Flow", "scope" parameter or "authorization_details" parameter in "Authorization Request", "proof" element or standard JWS for storing signature, "kid" element or self-descriptive "x5c" element to reference public keys in Verifiable Credentials).
- The level of Verifiable Credentials data structure: The DOME project aims to support JSON-based data structures of OIDC4VCI, OID4VP as it is required by ARF, but also



SD-JWT is required as well and selective disclosure of attributes shall be supported. Beyond JSON-based signed attributes of eIDAS 2.0 also transition from (or maintaining current) eIDAS 1.0-compliant data structures shall be supported. The current eIDAS 1.0-compliant data structures cover XML-based SAML Assertions or standard IETF RFC 5755 attributes certificates or CMS/PKCS#7-based ICAO Doc 9303 signed attributes in passport ID documents.

- The level of identifiers assigned to natural or legal persons: The DOME project aims to use partly internal identifiers for onboarding/registering natural persons and legal persons in different roles. In general, eIDAS 1.0-compliant systems support such processes (e.g. onboarding/registering User at an SP/RP such as a bank) based on government-assigned identifiers. Some of the attributes shall be supported in cross-border scenarios as well (8 natural person attributes, 10 legal person attributes and 2 for legal person representative natural person). The DOME project would support DID (W3C Decentralized Identifiers) and *organizationIdentifier* (ETSI EN 319 412) but eIDAS 1.0 requires to manage others as well (e.g. for legal persons beyond classic VAT Registration Number or Tax Reference also LEI, EORI, SEED, SIC and D-2012-17-EUIdentifier are defined).
- The level of signature creation: The DOME project aims to support JSON-based CSC (Cloud Signature Consortium) specification to access and use private keys to create qualified electronic signatures. This layer specifies the interface of a keystore (e.g. Cloud-HSM as server-side keystore and national eID cards or other PKI smart cards as client-side keystores) but the specification of a signature creation service - which shall be used by an EDIW - is on a higher level. This interface is described by ETSI TS 119 432 and it allows the use of both plain-old XML-based OASIS DSS-X standard and a newly defined extension to JSON-based CSC protocol.

3.1.2 Proof of participation

When verifying a Verifiable Credential/Presentation, we must address the following:

1. How do we determine whether or not we know that the **issuer** of the Verifiable Credential is a **participant** in the ecosystem where we are also participants (e.g., organisations onboarded in DOME)?
2. How do we determine whether or not we know that the **subject** of the Verifiable Credential is a **participant** in the concrete ecosystem where we are also participants (e.g., organisations onboarded in DOME)?

We propose to use a **Trusted Participant Registry** including the identities and associated metadata of all legal persons participating in the concrete ecosystem. In this document, and when there is no possibility of confusion, we will use the terms Trusted Participants Registry and Trusted Participants List as equivalent. The literature is not (yet) fully consistent and using the term List is closer to the eIDAS terminology, but there are many places where the term Registry is used. The Trusted Participant Registry is updated during the onboarding process of an entity and is managed by one or more collaborating trusted participants in the concrete ecosystem. Please note that this list is different from the EU Trusted List with the identities of TSPs authorised to issue digital certificates/seals in the EU.

There are different ways to implement the Trusted Participant Registry but in any case, the users of the Trusted Participant Registry should not be aware of the technology used to



implement it. The users of the Trusted Participant Registry just use an API to query the registry on verification, and the maintainers use a different API to register and update the registry.

This way, it is completely possible to use a mix of centralised and decentralised technology without the users noticing it. Or to migrate transparently from one technology to another depending on the requirements of the specific ecosystem.

Having said that, DOME will use a federated set of interoperable blockchain networks for the maintenance of the Trusted Participant Registry, providing a decentralised, hyper-replicated, efficient and resilient mechanism for querying the registry. Anyone can create a replica of the information using centralised systems if they wish.

The API is based on the one defined by [EBSI for Trusted Issuers Registry](#) of different types. For example:

GET /participants and ***GET /participants/{did}***

to get the list of participants or to check a given participant DID, respectively.

There are several other APIs to retrieve attributes/metadata associated with the participants, and APIs to maintain the list, used by the entity or entities responsible for the list. The full specification is later in this document.

We will follow this principle:

If something we need is already in EBSI, just use it. Otherwise define it trying to be as consistent as possible with EBSI, unless there is no chance to do so.

3.1.3 Proof of Issuing Authority

Given that anyone can have access to the technology needed to create Verifiable Credentials and anybody can issue credentials and digitally sign them with their eIDAS digital certificate, the problem is how a verifier knows that the Verifiable Credentials received from the subject have been issued by an entity which is entitled or authorised to issue that type of credential in the context of DOME.

3.1.3.1 Trusted Issuers Registries

The primary mechanism to solve this problem is the use of **Trusted Issuers Registries** (there may be several registries, one per domain or type of credential). In this document, and when there is no possibility of confusion, we will use the terms Trusted Issuers Registry and Trusted Issuers List as equivalent. The literature is not (yet) fully consistent and using the term List is closer to the eIDAS terminology, but there are many places where the term Registry is used. As long as the meaning is clear, we will use both interchangeably.

A Trusted Issuers Registry is a register of trusted public entities which can issue Verifiable Credentials belonging to a given domain or of a given type. In general, an entity must be first in the Trusted Participant List before it appears in a given Trusted Issuers Registry. In some ecosystems, the governance model of the ecosystem may allow participants to issue credentials of some type, without having to be included in a specific Trusted Issuers Registry. In this case, the governance model is effectively defining the Trusted Participants List as being also a Trusted Issuers Registry. However, in order to be general enough in the rest of this document we assume that the Trusted Participant List is different from the one or more Trusted Issuers Registries defined in the ecosystem.



A Trusted Issuers Registry includes the identifiers, public keys for verification of signatures and their accreditations in the form of Verifiable Credentials/Presentations from third parties, enabling the entity to issue credentials of a given type. All information in the registry is validated and signed by trusted legal entities of the corresponding domain (Conformity Assessment Bodies and third-party auditors).

Using Trusted Issuers Registries is the simplest mechanism, both for the verification of credentials and for the management of trusted issuers. However, in very complex ecosystems with many entities issuing credentials of different types, the management of Trusted Issuers Registries can be difficult to scale. For those ecosystems we can use the combination of Trusted Issuers Registries with the “chaining” of the signatures of Verifiable Credentials, like the certificate chaining used with traditional X.509 digital certificates:

- At the root of the trust hierarchy there is one or more Trusted Issuers Registries as described above, containing the primary trusted entities in the ecosystem.
- The entities in those Trusted Issuers Registries can issue special Verifiable Credentials to other entities, authorising them to be also Trusted Issuers, even if they are not included in a Trusted Issuers Registry. The signature of the special Verifiable Credential attests that the subject of the credential is explicitly authorised by the signer to issue a given type of credentials (usually a subtype of the parent type, but not necessarily; the specific rules have to be defined in the corresponding governance model for the domain/ecosystem). This mechanism can also be used recursively by those Trusted Issuers not in Trusted Lists if we need several levels in the hierarchy, though usually two or three layers (including the root Trusted Issuers Registry) should be enough to handle large ecosystems.

There is a trade-off in choosing one mechanism or the other. Having all Trusted Issuers in one or more Trusted Issuers Registries makes verification very simple: the verifier of a credential just checks once in the Trusted Issuers Registry corresponding to the type of certificate. Checking a credential using the chained mechanism is more complex: the verifier has to check the chain of signatures for the relevant Verifiable Credentials until it reaches an issuer which is in a Trusted Issuers Registry for the domain (or if no issuer is in any Trusted Issuers Registry, then the Verifiable Credential should be rejected).

Using only Trusted Issuers Registries makes it easier to know at a given point the whole set of Trusted issuers in the whole ecosystem (or on the domain covered by a given Trusted Issuers Registry). This is very difficult to know with the chained mechanism.

In a given implementation, the mechanisms can be combined and are not exclusive or all-or-nothing in an ecosystem. Depending on the requirements/complexity of a domain in an ecosystem, one domain can use just Trusted Issuers Registries while another domain can use a chained mechanism. It is even possible to start with only a Trusted Issuers Registry and transition seamlessly to the chained mechanism if the domain complexity grows beyond some limit, decided by the governance rules of the domain.

3.1.3.2 The List of Trusted Lists (LOTL)

In a complex ecosystem like DOME there will probably be more than one Trusted Issuers List, each list specialised in some business, organisational or technical domain with its own set of specific Verifiable Credentials and with possibly different governance models for updating the information in each list.



Having different specialised Trusted Lists is good for decentralisation of the ecosystem, as each domain or subdomain represented by a list can be managed by the entity or entities which are responsible for the Trust Anchors in the lists.

However, it complicates the verification process of Verifiable Credentials, especially when an entity requires a combination of several credentials of different types in order to implement a given business process. The Verifier then has to know the location of different lists that are required to locate the different signers of the credentials.

One way to simplify the process is to implement a directory of lists, or using terminology coming from eIDAS, a “List Of Trusted Lists” like [the one operated by the European Commission pointing to the trusted lists of all Member States](#).

In a similar way, DOME will make publicly available (that is, not restricted for reading) a List Of Trusted Lists (LOTL), pointing to all the Trusted Issuers Lists recognised in the ecosystem. This list will be available in a signed or sealed form suitable for automated processing and will include required information and metadata to be useful to verifiers.

The LOTL will be registered in the federated blockchain platforms in DOME, and in this way it is automatically hyper-replicated in a trusted way to all the entities participating in the ecosystem that run a blockchain node in any of the federated blockchain networks.

3.2 Onboarding of DOME participants

The description of the onboarding process is divided into two distinct phases:

1. Identification and verification of future participants when they sign up with DOME
2. Registration of those new participants in the
 - a. Trusted Participant List associated with DOME participants
 - b. Trusted Issuers Registry for registering them as trusted issuers of verifiable credentials meaningful for the DOME business logic (e.g., app/provider, customers, federated marketplace)

The Trusted Participant List and Trusted Issuers Registry are replicated across the supporting federated blockchains, using the federated trust framework mechanisms described later in the document.

We describe below each of the two phases.

3.2.1 Identification and Verification of future participants

This phase refers to a process which precedes entering into a business relationship with a new participant, which has to be a legal person (at this moment we do not address onboarding of natural persons).

In general, the onboarding process is one of the less digitised and more diverse, with different implementations depending on the sector of activity and local regulation. Even within the same sector the actual implementation of onboarding processes for different companies can vary considerably.



Onboarding of participants in an ecosystem may also present differences depending on the specific ecosystem. This chapter presents an approach based on eIDAS digital certificates that can facilitate in the EU area a **fully digital and automated cross-border onboarding process and compliance with KYC** (Know Your Customer) requirements.

Onboarding of a new participant in the ecosystem is a critical activity and proper identification of the new participant at this stage affects very much to the level of trust of the whole onboarding process and so the level of trust that all the other participants in the ecosystem can place on the identity of the new participant. The objective in DOME is to provide a reasonable balance between convenience and level of legal certainty and security of the electronic transactions involved in the onboarding process.

To facilitate the descriptions later in this document, we reproduce here some relevant definitions taken from the eIDAS Regulation:

'Electronic identification' means the process of using person identification data in electronic form uniquely representing either a natural or legal person, or a natural person representing a legal person.

'Person identification data' means a set of data enabling the identity of a natural or legal person, or a natural person representing a legal person to be established.

'Authentication' means an electronic process that enables the electronic identification of a natural or legal person, or the origin and integrity of data in electronic form to be confirmed.

'Relying party' means a natural or legal person that relies upon an electronic identification or a trust service;

The onboarding process presented in this chapter is just one of the possible options that can be implemented, but it should be the main one supported in the EU region given its advantages.

3.2.1.1 Legal basis

The eIDAS Regulation (EU) 910/2014 (hereafter denoted as eIDAS) is a major step towards building the EU Digital Single Market (DSM) as it provides a predictable regulatory environment for the cross-border recognition of electronic identification (eID) and electronic trust services by Trust Service Providers (TSPs). eIDAS may facilitate to meet the legal obligations, concerning security, know-your-customer, strong authentication of parties and interoperability.

For the purposes of this chapter, this phase of the onboarding process consists of the following logical phases (which can be combined in a single step when performing them in a digital way):

- **Application.** Pre-on-boarding phase, addressing the act of applying to become a participant. In this phase, the applicant provides the required identity and KYC attributes for verification and collection.



- **Verification.** The verification phase determines whether the expected requirements and mechanisms used to perform verification of attributes are met. It can be divided into 3 steps:
 - Authenticity check of documents, to determine that the document can be considered a trustworthy source of information such as for identity attributes.
 - Identity check of the applicant, by comparison of the bearer of the document against the owner of the document.
 - Anti-fraud check, to determine that the document is not used in fraud-related activities and it belongs to a living person or existing legal entity; and that the applicant is not involved in fraud activities, not under sanctions or considered a PEP ([Politically Exposed Person](#)).

Depending on the sector of activity and its legal and regulatory requirements, a complete onboarding process requires the following common due diligence measures:

- **Identification of legal person** on the basis of documents and data submitted; and verification of the submitted information on the basis of information obtained from a reliable and independent source;
- **Identification and verification of the legal representative** and the right of representation;
- **Identification of the beneficial owner**, based on information provided for onboarding or obtained from another reliable and independent source; and
- Obtaining information on the **purpose and nature of the business relationship or transaction**.

In this chapter we focus only on the identification of the legal person and in the identification and verification of the legal representative, and assume that once this is done the remaining documentation can be provided and verified in a trusted, digital and automated way to complete all remaining required steps in the specific onboarding process for the ecosystem.

In DOME the main mechanism to provide those documents is to submit them in the form of Verifiable Credentials since this makes the process much easier and reduces manual work and possible error: the onboarding entity can check if the corresponding credentials have been signed by trusted issuers.

Types of digital certificates in the EU

The relevant types of certificates for the onboarding process are issued to natural persons, legal persons and natural persons representing a legal person (as described in article 3(1) of eIDAS for the case of *representation*).

Based on the regulation, some TSPs (Trust Service Providers) in the EU provide several types of digital certificates for digital signature/seals:

- **Natural Person** certificate for electronic signatures
- **Legal Person** certificate for electronic seals



- **Natural Person as Legal Entity Representative** certificate for electronic signatures

Note: electronic signature vs. electronic seals⁵

Electronic signature: An electronic signature is a data in electronic form which is attached to or logically associated with other data in electronic form and which is used by the signatory to sign, **where the signatory is a natural person.**

Like its handwritten counterpart in the offline world, an electronic signature can be used, for instance, to electronically indicate that the signatory has written the document, agreed with the content of the document, or that the signatory was present as a witness.

Electronic seal: An electronic seal is a data in electronic form, which is attached to or logically associated with other data in electronic form to ensure the latter's origin and integrity, **where the creator of a seal is a legal person** (unlike the electronic signature that is issued by a natural person).

In this purpose, electronic seals might serve as evidence that an electronic document was issued by a legal person, ensuring certainty of the document's origin and integrity. Nevertheless, across the European Union, **when a transaction requires a qualified electronic seal from a legal person, a qualified electronic signature from the authorized representative of the legal person is equally acceptable.**

Some terminology

Not all Member States implement at this moment the certificate for a Natural Person as Legal Entity Representative, but the onboarding process described below takes advantage of it when it is available for a participant initiating the onboarding.

In this way, the onboarding process is prepared for the future, because given that there are different cases of representation, the eIDAS Technical subgroup has been requested by the eIDAS Cooperation Network to amend the technical specifications to include all the cases of representation (see section 2.8. NATURAL AND LEGAL PERSON REPRESENTATIVE from eIDAS SAML Attribute Profile V1.2., 31 August 2019). It is expected that in the near future most TSPs will start issuing those digital certificates that simplify and streamline enormously the onboarding processes, not just for this specific use case but for any type of use case in the economy.

To facilitate the explanation below, we will use the following terminology:

- **NP:** Natural Person with a *certificate for electronic signature*.
- **LE:** the Legal Entity that is being onboarded, with a *certificate for electronic seal*.

⁵ Source: European Commission eSignature FAQ (<https://ec.europa.eu/digital-building-blocks/wikis/display/DIGITAL/eSignature+FAQ>)



- **LER:** the Natural Person as Legal Entity Representative with a *certificate for electronic signature* when acting as legal representative of a legal entity.

The above concepts have a one-to-one relationship with the legal entities in eIDAS. However, we need a little bit more flexibility with respect to the person/employee that can initiate and drive the onboarding process in the ecosystem. Even in a big company there are not many employees that have the power of legal representation. We define here an additional concept that is being used already in many other contexts:

- **LEAR:** Legal Entity Appointed Representative, a natural person that has been nominated (appointed) by a legal representative to act on behalf of the organisation to perform a limited set of processes, in our case the onboarding process (and maybe some additional tasks).

Instead of using traditional digital certificates, we require that the LEAR receives a special type of Verifiable Credential called **LEARCredential** with proof that the person has the power to represent the legal person in some limited capacity. This credential is described later in this document.

This concept of LEAR is very similar to the equivalent one for how organisations interact with the European Commission to perform certain tasks on behalf of their organisation, as part of its participation in EU funded grants, procurements and prizes that are managed via the EU Funding & Tenders Portal. To illustrate further, see below the description of LEAR from the Commission.

LEAR appointment and validation⁶

Parallel to the validation of your organisation, you will be requested by the Central Validation Service to appoint your Legal Entity Appointed Representative (LEAR).

This must be done by a legal representative of your organisation with the necessary legal authority to commit the organisation for this type of decisions (e.g. typically CEOs, rectors, Director-Generals, etc. always in accordance with the statutes of your organisation).

The LEAR role, which can be performed by any member of the organisation (typically from the central administration), is key. They are formally nominated to manage your organisation's use of the Portal and thus bear the final responsibility for all your actions in the Portal. Once validated, they will be responsible for:

- *keeping an overview of all the proposals/projects/contracts your organisation is involved in*
- *managing all the legal and financial information about your organisation*

⁶ Source: European Commission (<https://webgate.ec.europa.eu/funding-tenders-opportunities/display/OM/LEAR+appointment+and+validation>)



- *managing the access rights at organisation-level (and read-only access at project-level)*
- *appointing the persons which will be able to electronically sign grants/contracts (Legal Signatories — LSIGNs) and cost claims/invoices (Financial Signatories — FSIGNs).*

We have to make a detailed definition of the LEAR responsibilities in our context, and specify that instead of a signed PDF we require a signed Verifiable Credential with the proper content.

3.2.1.2 The LEARCredential

The objective of the onboarding process in an ecosystem is to identify and register a legal entity as a new participant. However, in most cases the process is initiated and driven by a natural person.

To provide higher legal certainty and higher security, the onboarding process requires that **the natural person driving the process should be either a legal representative of the participant or a natural person that has been delegated by a legal representative** at least the powers required to perform the onboarding process on behalf of the legal entity.

The onboarding process is based on a special type of Verifiable Credential that we will call LEARCredential, given its purpose. We require that the user driving the process is a LEAR **holding and controlling** a LEARCredential.

The LEARCredential can be generated in different ways:

1. Using the LE certificate. The person controlling the LE certificate issues a Verifiable Credential to a natural person which typically is an employee of a department in charge of managing the onboarding processes and the relationship with a given ecosystem. The Verifiable Credential includes a description of the actual powers that are being delegated (the required ones have to be defined by the ecosystem). The Verifiable Credential is sealed with the digital certificate of the LE. This VC is then called a LEARCredential and the person controlling it can use it to authenticate in the onboarding process and act as LEAR.
2. Using the LER certificate. The main difference with the above is that in this case the Verifiable Credential is signed with a LER certificate. This tends to be easier, especially in big companies because there are normally more than one LER depending on the company structure. As a special case, the LER can issue a LEARCredential to herself and then become a LEAR. This can be used when the LER wants to be the one performing the onboarding process.
3. Using a trusted entity before onboarding. When neither LE nor LER digital certificates are already available in the company, a TSP or other trusted entity accepted by the ecosystem could generate and sign the Verifiable Credential directly, instead of generating a LER certificate.



The main difference with the previous scenarios is that the LEARCredential is signed by a trusted entity and that it has to be involved before the onboarding process starts, making the process somewhat more cumbersome and less self-service.

In any case, the LEARCredential replaces its analog counterpart with a more efficient, machine-interpretable version. Following the example of the LEARCredential for the Funding & tender portal of the EC, the LEARCredential would replace the natural language letter that has to be signed: [LEAR APPOINTMENT LETTER](#).

The major problem here is to what level the natural language description of the roles and duties of the LEAR and delegation of those will be described in a formal language, versus simply embedding a secure pointer to somewhere where the actual natural language description is stored. This is an important subject for access control.

3.2.1.3 The identification and verification phase

The onboarding service of a Data Space can act as a Relying Party and use the LEARCredential and the mechanisms for Identification and Authorisation described in this document to properly identify the legal person involved, the natural person performing the onboarding and that the natural person has the powers of representation.

This can be done fully automated and without requiring a previous relationship among the new participant and the onboarding service, enabling self-onboarding.

Once this step is performed, the LEAR can provide additional documents (ideally as Verifiable Credentials) to complete the onboarding process or perform other required validations.

3.2.2 Registration of new participants in the Trusted Lists (DigitITS)

After the legal entity has got the **LEARCredential**, the entity must be added to the **Trusted Participant List** stored in the federated blockchain networks of DOME. For this purpose, it is good to have a publicly permissioned Blockchain as it is closer to regulatory compliance than a public non-permissioned one and still has the advantages of hyper-replication, decentralisation, immutability, no single points of failure and no vendor lock-out. Furthermore, it is more scalable and energy efficient.

The modification of this list must be access controlled. For that purpose a similar approach to the one followed by the **Trusted Issuers** of Credentials can be followed as this access can be controlled using Credentials too. This approach can benefit from following a similar model than the one described in [EBSI's proposal for issuer's accreditations](#). In that scenario our problem is converted to a mere Proof of Issuing Authority problem to be solved. An entity that is considered as a trusted entity that can modify this list, which we will name as a Trusted Participant Credential Issuer, must comply with the following:

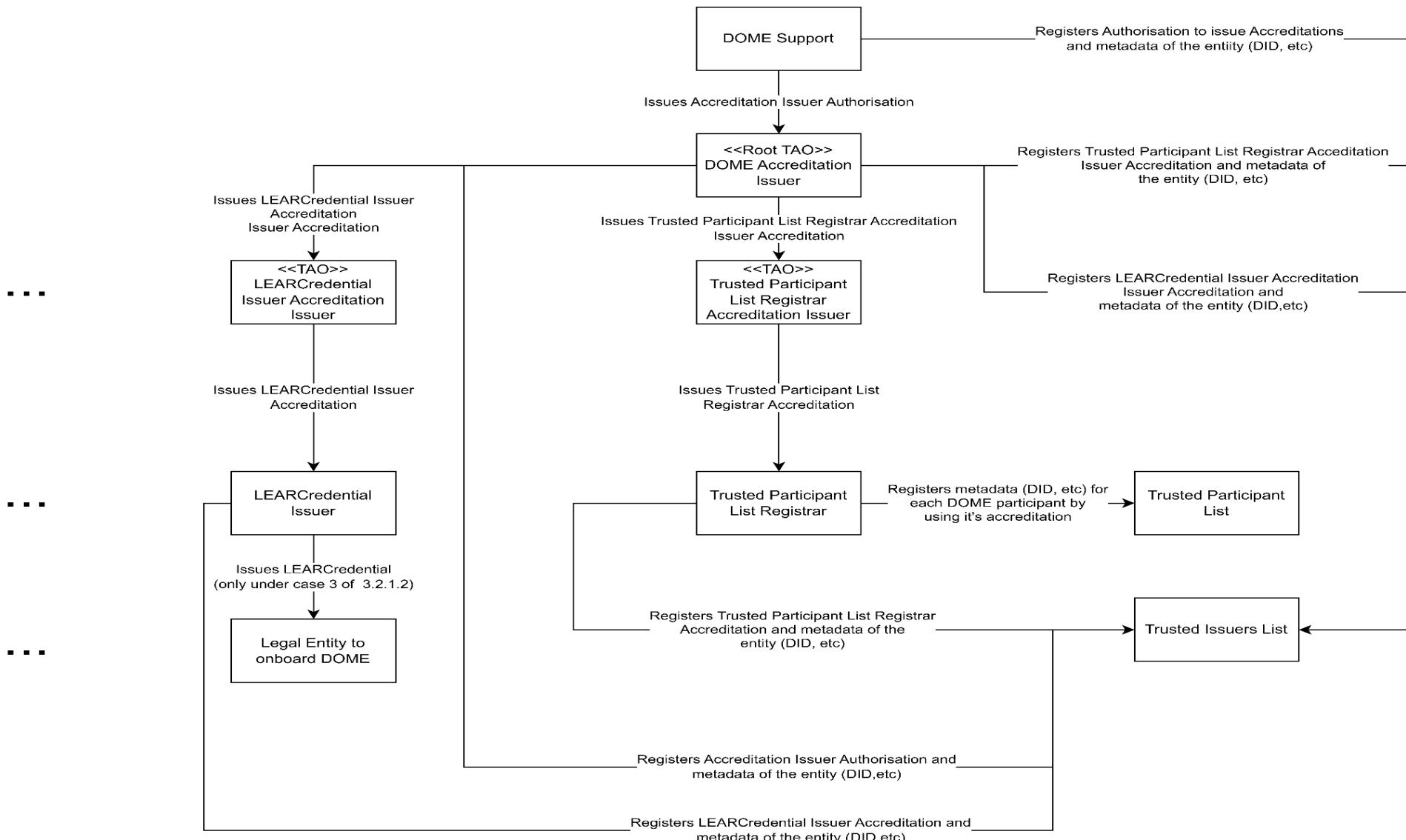
- It has to be added to the **Trusted Participant List** itself.



- It has to have a **Trusted Participant List Registrar Accreditation** issued by a **Trusted Participant List Registrar Accreditation Issuer**.

For those things to happen there must be a top level entity authorised to issue accreditations to issue certain accreditations to be able to distinguish between entities allowed to issue some accreditations from others that can only issue accreditations about a different domain or different credentials. That entity will be called a **TAO** or **Trusted Accreditation Organizations**. A certain domain can choose to use one or more Trusted Lists for it's Trusted Issuers or a chained approach like the one referred to as "chaining of Verifiable Credentials" in section 3.1.3. The following figure illustrates the general approach described previo



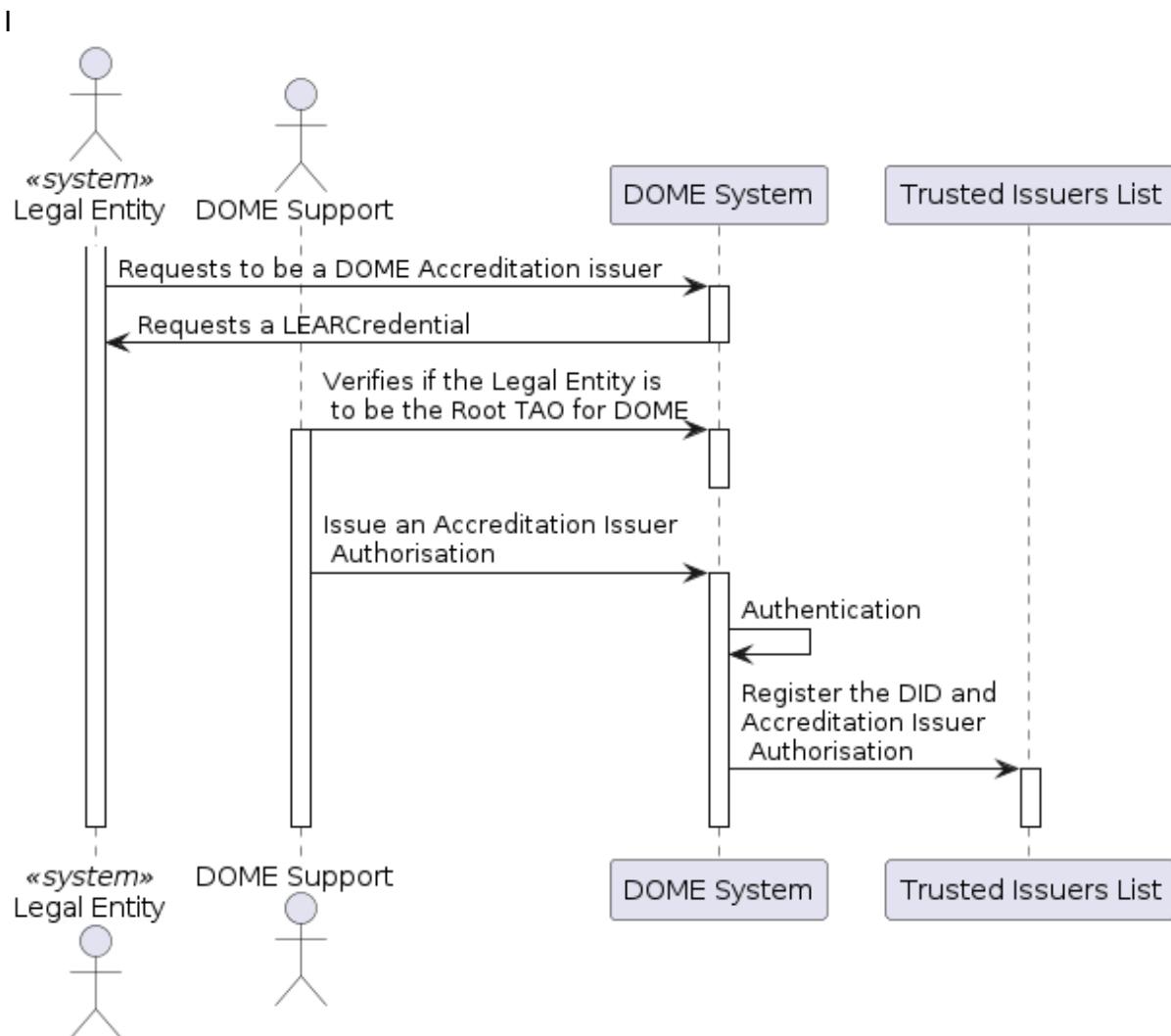


3.2.2.1 Root TAO registration

At the top of the hierarchy we have an Accreditation Issuer acting as a **Root TAO or Root Trusted Accreditation Organization**), a concept defined by EBSI which marks that issuer as the root of trust for the ecosystem in the sense that it can issue all of the accreditations of the ecosystem by using its **Authorisation** issued by DOME Support. The **Accreditation** credential is a special credential that enables issuance of a certain Verifiable Credential(s), including other Accreditations. The Accreditations for each type of Issuer allowed and the Authorisation for the Root TAO are stored in the Trusted Issuers Registry of DOME, along with other metadata like DIDs, public keys of the Issuers etc. The Root TAO Authorisation is registered and issued by the DOME Support to be able to establish the root trusted entity of the ecosystem. The Root TAO can use the Authorisation to issue accreditations to others. All accreditations are stored in the Trusted Issuers Registry so they are not needed after the onboarding performed by the different Trusted Issuers in which it is registered. There is a unique Trusted Issuers Registry for issuers of credentials not belonging to a specific participant domain. The Accreditations to accredit each credential are registered so a unique list is possible and we don't need different lists for different types of issuers.

The process to onboard a Root TAO to the DOME ecosystem is as follows:



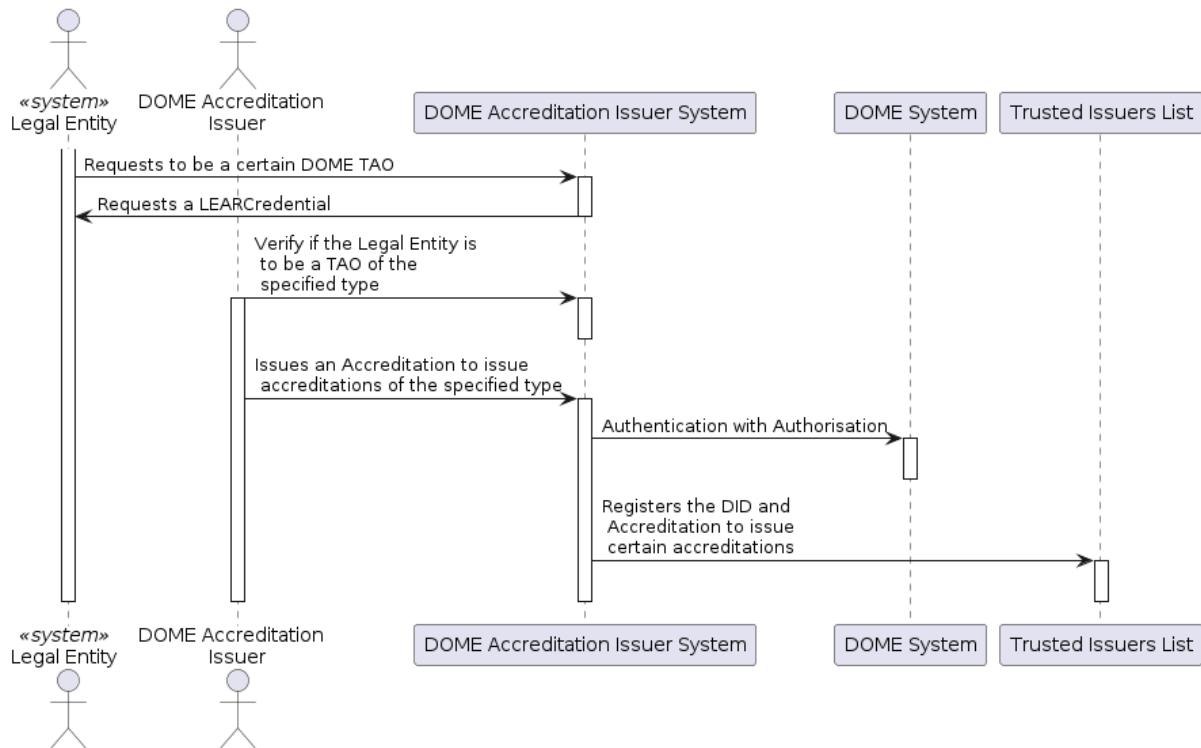


Note that in this case there is a manual process to verify that the Legal Entity represented by the LEAR is to be the Root TAO for DOME and issue the Authorisation to make the process as secure as possible given that this kind of issuers establish the main truth hierarchy.

3.2.2.2 TAO registration

The Root TAO is able to accredit new legal entities to issue the different top-level Credentials used in DOME which are called TAOs as said before. The Accreditation to accredit issuers of certain credentials is given to a certain TAO by the Root TAO, which registers the accreditation too for a more secure onboarding. Our TAOs could be something like a **LEARCredential Issuer Accreditation Issuer**, that can accredit legal entities to issue LEARCredentials, a **Trusted Participant List Registrar Accreditation Issuer** that gives accreditations that enable to modify the Trusted Participant List or a **TAO that manages the issuance of accreditations for new DOME domains**. We can add as many intermediate TAOs as we want before issuing accreditations to a non-TAO legal entity which allows for a diverse trust hierarchy.



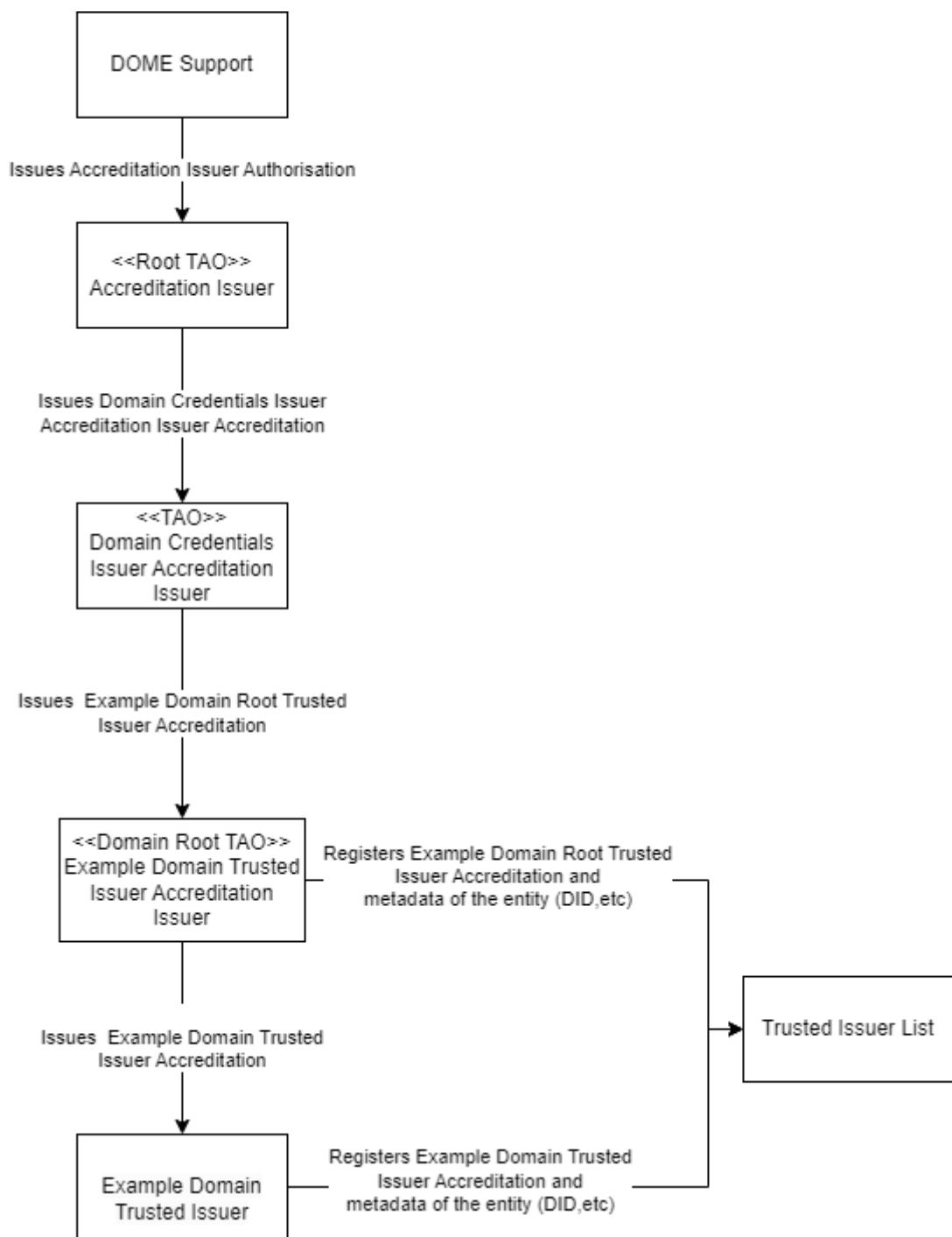


Note that in this case there is a manual process to verify that the Legal Entity represented by the LEAR is to be a TAO for DOME and issue the Accreditation to accredit issuers of certain credentials for the same reasons as in the Root TAO.

3.2.2.3 Domain-specific Trusted Issuer

The first non-TAO legal entity, at least in the general sense for a DOME TAO, in the case of Issuers of a certain domain Credential is considered to be the Root TAO for that particular domain in the sense that it can issue each and every accreditation for credential issuance of Verifiable Credentials. This is possible because it is in the general Trusted List of DOME and can act as a trusted source. With this approach a certain domain can choose to use the trust chain approach or plain Trusted List(s) when a domain considers one or another way to be more beneficial for their use case. A high level view of this case can be seen in the following figure:

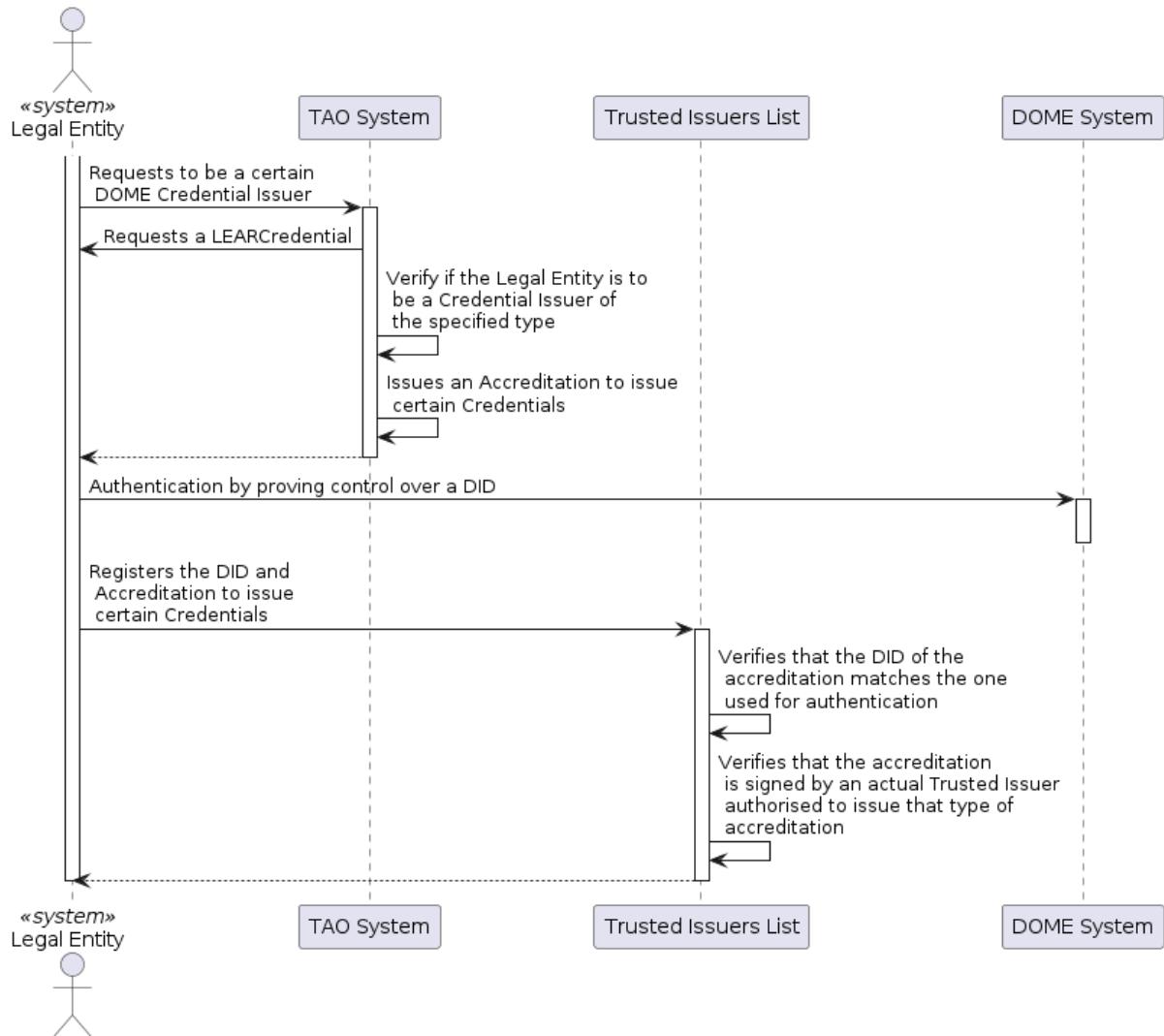




3.2.2.4 Trusted Issuers registration

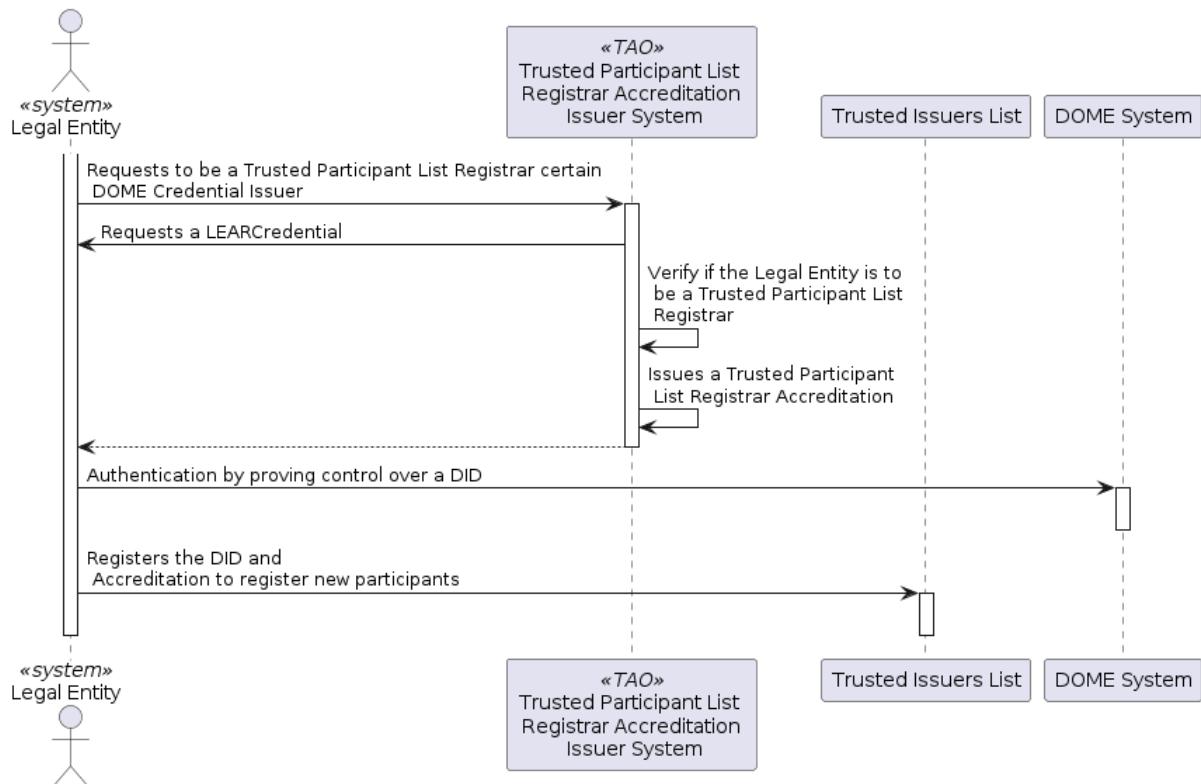
In the figure can be seen that a Legal Entity considered as valid by a TAO is considered to be accredited as a certain Trusted Issuer that can self register in the Trusted Issuers Registry after its accreditation is issued. Since the Accreditation must include the Legal Entity DID, by proving that it has control over that DID it can be authorized to register that Accreditation linked to that DID in the Trusted Issuers Registry.





3.2.2.5 Trusted Participant Registrar registration

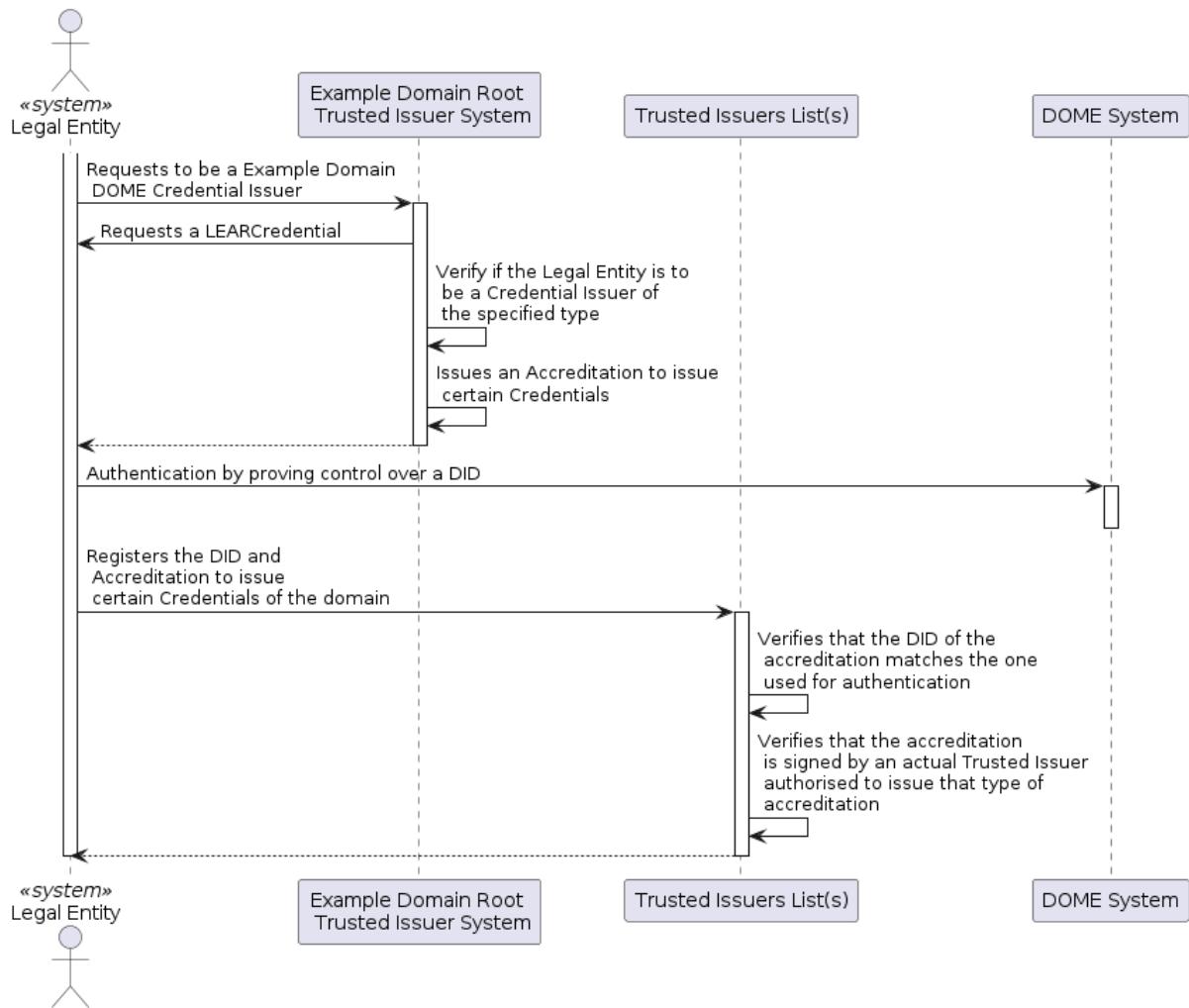
The registration of a Trusted Participant Registrar, which is a Legal Entity that can give new accreditations to Legal Entities to be able to write in the Trusted Participant List, is the same as for a Trusted Issuer. The only difference between a conventional Trusted Issuer and this special type of issuer is that the former issues credentials but this one does not, it only works as an authorization mechanism to access the Trusted Participant List. A sequence diagram detailing this registration can be seen in the following figure:



3.2.2.6 Registering of a Trusted Issuer of a domain using Trusted List(s)

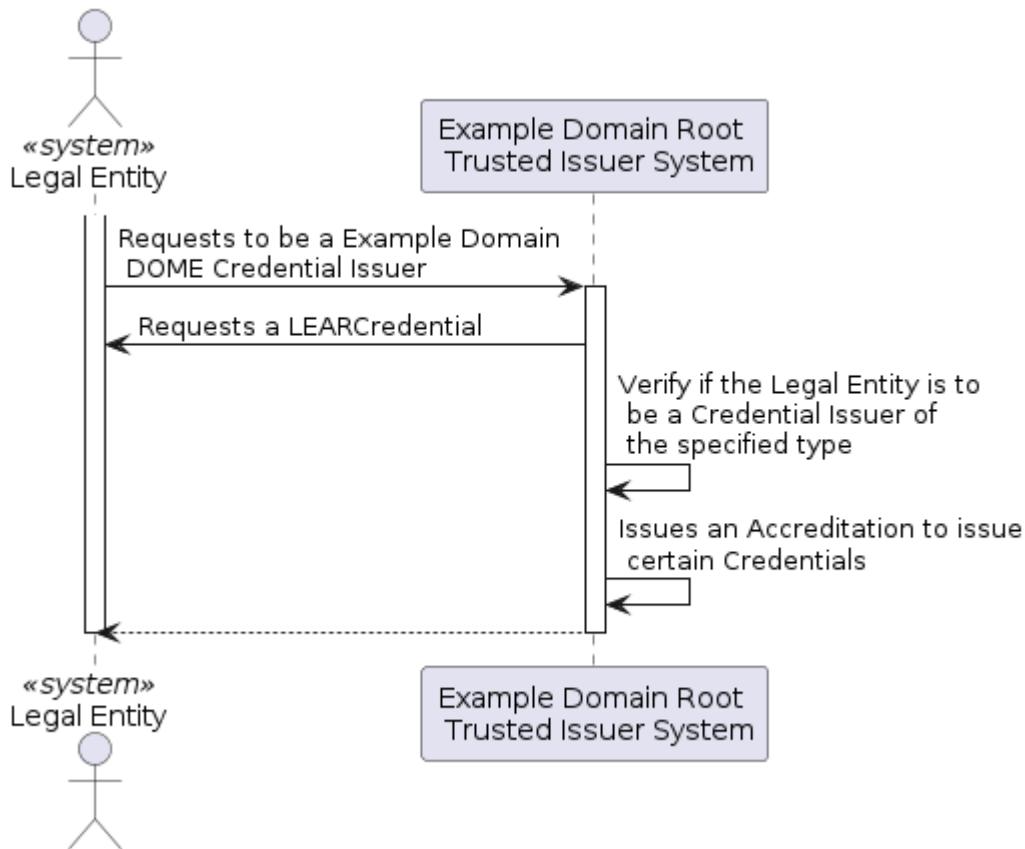
This case is identical to the case of registering a generic DOME Trusted Issuer. It is described in the following figure:





3.2.2.7 Registering of a Trusted Issuer of a domain using a trust chain

The idea of a trust chain approach for Verifiable Credentials is to be able to verify the issuer of a Credential without that issuer being in any kind of Trusted List. For that the issuer of a credential to be verified must present an Accreditation attesting that the issuer is allowed to issue that kind of Credentials. Since the Accreditation is issued by a Trusted Issuer in the DOME Trusted List it can be verified that the issuer of the Credential is able to do so according to the established trust in the ecosystem. This means that the process is similar to the registration using Trusted Lists but without registering the Accreditation itself. The following figure describes this approach:



3.3 Authentication and Authorization with Verifiable Credentials

3.3.1 Introduction

In a typical B2B ecosystem, the agreements between a product/service provider and the consumer are formalised among legal persons (also known in this document as 'organisations').

However, when interacting in the context of execution of the agreement, typically other entities are acting on behalf of the legal persons both for the consumer organisation and for the service provider organisation.

For example, an employee of the consumer organisation may start, on behalf of his employer, an interaction with the provider organisation to perform a process needed for the procurement and provision of a service. In this case, the provider organisation should authenticate the natural person and apply appropriate access control policies to ensure that the natural person is really entitled by the consumer organisation to perform the intended process.

Another example is when a device, server machine or a service controlled by the consumer organisation interacts with a service provider in an automated way (“controlled” means in this context that the device, machine or service is operating under the responsibility of the legal person). Before allowing the entity to access protected resources, the provider organisation should authenticate the entity and apply appropriate access control policies to ensure that the entity is really entitled by the consumer organisation to perform the intended process.

In the context of authentication and access management, the legal person acting as consumer organisation will be named **Participant**. The digital representation of an entity acting on behalf of a Participant is referred to as a **Principal**. In Human-to-Machine (H2M) scenarios, the **Principal** is a natural person (typically an employee or contractor), while in Machine-to-Machine (M2M) scenarios the **Principal** can be a device, an application or any other automated entity without legal personality under eIDAS. However, when the context is clear and we need to refer to eIDAS legal terms, we will use the term **user** (defined in the Glossary) to denote a natural or legal person, or a natural person representing a legal person using a wallet to authenticate.

The legal entity acting as service provider and performing authentication and access management will be called **Relying Party**.

We describe here a mechanism to use Verifiable Credentials to perform Authentication and Access Control leveraging the eIDAS trust framework and using advanced or qualified signatures and seals to provide a high level of legal certainty and especially enjoying the presumption of non-repudiation provided by those eIDAS signatures.

We focus here on **legal persons** and natural persons or machines acting on behalf of the legal person with its authorisation. The identification and authentication of **natural persons** acting on their own (citizens) is out of scope (except when acting as representatives of the legal person, see below). Whenever eIDAS2 and the EUDIW (European Digital Identity Wallet) are ready, we will support that mechanism for identification and access control.

The mechanism described here allows a Relying Party receiving a Verifiable Credential from a principal engaging in an authentication process to perform the following verifications:

Authentication

- **Non-tampering.** The Relying Party can verify that the credential was not tampered with since it was generated, thanks to the digital signature of the credential.
 - **Binding the Principal with the subject inside the credential.** The Relying Party can verify that the principal presenting the credential is the same principal identified as the subject inside the credential by using the cryptographic material inside the credential that was embedded in a secure and private way during the credential issuance process. See section [Authentication requires a VerifiableID](#) for more details.
- The degree of trust in this verification depends on the degree of trust that the



Relying Party puts on the processes that the Participant uses to issue the credentials.

However, as described below, the usage of eIDAS signatures to sign the credential makes the Participant legally liable for any problems in the binding of the identities of the holder and subject of the credential. Specifically, compliance to the eIDAS specifications and regulation for electronic seals and signatures provides **presumption of non-repudiation and the burden of proof in the courts rests on the Participant, not on the Relying Party**. This is important because it provides the incentives in the proper places: the Participant is incentivised to create the credential in the proper way, and the Relying Party is incentivised to perform its due diligence in verifying the credential. By using the eIDAS signatures, no party has to assume the burden if the technical validity of the signature is disputed in court. Only the contents of the credential can be under discussion. See section [Signature of the Verifiable Credentials](#) for more details.

In the proposal of this document, the verification is enabled by embedding inside the credential a public key corresponding to a private key that was generated by the principal during the issuance process. See below for the details, including an optional mechanism by embedding an existing eIDAS certificate issued by a QTSP when the principal is a natural person or a legal person (different from the Participant).

- **Binding of issuer with a real-world identity.** The Relying Party can verify that the credential was issued by somebody controlling a private key associated with a given real-world identity. The verification is enabled because the Verifiable Credential is sealed with an eIDAS qualified certificate for electronic seals issued to the Participant by an EU Qualified Trusted Services Provider (QTSP), and the signature of the credential is an advanced or qualified electronic signature using the format JAdES (JSON Advanced Electronic Signature) as defined in the eIDAS regulation.
Of course, we also support electronic signatures with qualified certificates for electronic signatures issued by QTSPs to **natural persons acting as representatives of legal persons**. This verification ensures that an organisation with a real-world legal identity under the eIDAS Trust Framework has signed the credential being presented by a natural person identified as the subject of the credential. This verification **does not say anything about whether the organisation is a participant** in a given Data Space or not. See next point for such verification.
- Verify that the real-world identity is a Participant. The Relying Party, using the unique identifier associated to the eIDAS certificate used to sign the credential, can check if the issuer of the credential is a participant in a given Data Space by looking for that unique identifier in the Trusted Participant List managed by that Data Space. This unique number is the unique organizationIdentifier inside the certificates for legal persons and certificates for natural persons acting as representatives of legal persons, defined in [ETSI EN 319 412-3 V1.2.1 \(2020-07\) - Electronic Signatures and Infrastructures \(ESI\); Certificate Profiles; Part 3: Certificate profile for certificates issued to legal persons.](#)



Any legal person that can already engage in electronic transactions in the internal market has an eIDAS certificate including such a unique identifier, which is used in all types of legally-binding signatures like invoices, contracts, etc.

We assume here that the onboarding process of each Data Space / Ecosystem does not invent a new "local-only" identifier but uses the one that is already valid for electronic transactions in the internal market. Or at least, that the global eIDAS unique identifier is associated with whatever private identifier is invented by the Data Space.

We assume that the eIDAS unique identifier is used during onboarding to update a given Trusted Participant List with the identity of the new participant. That identity is composed of the unique identifier in the eIDAS framework and any additional attributes that may be relevant for the ecosystem. See below for the structure of the unique eIDAS identifier.

Access Control

- **Binding the Participant to the delegated powers to the Principal.** The Relying Party can verify that the legal person (the Participant) or a representative of the legal person (in eIDAS terms) has delegated a specific set of powers to the subject identified inside the credential (who is the same person holding and presenting the credential, as described above in 'authentication').

This is enabled because the credential includes a **set of claims that state in a formal language the delegated powers**, and the credential has been sealed/signed using an eIDAS certificate using the eIDAS advanced/qualified electronic signature format, described in more detail later in this document. Alternatively, the credential can include a pointer to an external document (which could be another credential) describing in detail the delegated powers. Thanks to the usage of eIDAS signatures, the credential is effectively a legal document with the same legal validity as any other document in any format supported by the eIDAS signature scheme. The Relying Party can verify (and prove in court if needed) with a high level of legal certainty that the legal person issuing the credential (the Participant) has declared that the holder/subject of the credential has the powers stated inside the credential (or linked by the credential).

Actually, the entity identified as a subject in the credential, and that is receiving the delegated powers, is not restricted to be a natural person and can be of different types. Anything that can be expressed in paper or PDF format can be also expressed in the Verifiable Credential format. For example, it can be an employee, a customer or a machine or device under the responsibility of the legal person signing the credential.

- The Relying Party can use the claims mentioned above, expressed as a formal language (e.g. ODRL), to perform access control. The specific mechanism is not yet detailed in this document. For example, the credential can just specify the role(s) of the entity identified in the credential (e.g. 'Finance Administrator') and the Relying Party can evaluate a set of arbitrarily complex policy rules associated to the role(s) for access control.

Alternatively, the credential could specify more complex rules which the



Relying Party can combine if needed with additional policy rules to make the final decision.

To achieve all of the above, we define specific types of Verifiable Credentials with a format and a mechanism which specify:

1. a way to sign the credentials;
2. a way to bind the identity of the holder of a credential to the identity of the subject inside the credential;
3. and a way to embed authorisation information in the credential describing the powers that the legal entity has delegated to the subject inside the credential.

The mechanisms described here can be used to generate credentials to employees, contractors, customers and even to machines and devices acting on behalf of an organisation. The schema is identical, except the issuance process is probably a little bit different and the roles and duties embedded in the credentials have different legal implications (but always in line with the legal framework, for example with the Consumer Protection Directive).

3.3.1.1 Signature of the Verifiable Credentials

We use **qualified certificates for electronic seals** provided by eIDAS Qualified Trust Services Providers (QTSPs) to seal credentials with advanced electronic seals (AdESeal) or with qualified electronic seals (QESeal) with the JSON Advanced Electronic Signature format (JAdES).

We also use **qualified certificates for electronic signatures** issued by a QTSP to authorised representatives of the legal person, to sign credentials with advanced electronic signatures (AdES) or with qualified electronic signatures (QES) with the JSON Advanced Electronic Signature format (JAdES).

This mechanism for sealing/signing the Verifiable Credentials has the following properties:

Provides high assurance of the identity of the creator of the credential.
The seals/signatures provide **high assurance of the identity of the creator of the seal**. For example, it will be difficult for a malicious user to get a qualified seal certificate in the name of a company, because the QTSP will be responsible to check that such a seal is issued to the persons representing the company and not to unauthorised persons.

Enables authorised representatives of the legal person to act on behalf of the legal person.

The mechanism provides **high legal predictability**, including for the qualified electronic signature the benefits of its **legal equivalence to handwritten signatures**. As stated by the eIDAS Regulation, when a transaction requires a qualified electronic seal from a legal person, a qualified electronic signature from the authorised representative of the legal person should be equally acceptable. It is possible to certify elements that are bound to the signatory such as a title (e.g. Director), a link with its employer, etc. Because the QTSP is trusted for verifying the information it certifies, the Relying Party can get a high level of confidence in such information conveyed with the signature through the signatory's certificate.



In this way, the Relying Party receiving Verifiable Credentials can have the same legal certainty than with any other document in other formats (e.g. PDF) signed with AdES or QES signatures or with handwritten signatures (in the case of QES).

Provides a high level of legal certainty and interoperability. Any basic signature benefits from the non-discrimination rule, which means that a Court in an EU Member State cannot reject it automatically as being invalid simply because they are in electronic form. However, their dependability is lower than that of an AdES/AdESeal or QES/QESeal because the signatory may be required to prove the security of the technology being used if the validity of the signature is disputed before a court. This requires significant costs and efforts that could be avoided with relative ease by opting for the more established and standardised advanced and qualified signature solutions. It may also be the case that the relying parties have no applications or tools to validate such signature, when not based on standards; in such a scenario, the signature may be legally valid and technologically robust, but of limited use (see [[ENISA-QES](#)] and [[ENISA-QSEAL](#)]).

For these interoperability reasons, QES/QESeal that are based on recognised EU standards are preferable unless the parties operate purely in a local context where the acceptance and usability of the chosen signature solution is sufficiently certain. Beyond the technical interoperability, the eIDAS Regulation also ensures the international recognition of electronic signatures, and not limited to the EU.

3.3.1.2 A taxonomy of Verifiable Credentials

The objectives defined in the introduction can be achieved in different ways. We define here an approach that standardises the detailed mechanisms so it can be adopted easily by anyone who wants to obtain the associated benefits. If the approach is adopted in an ecosystem, it provides a high level of interoperability among the participants in the ecosystem, reducing also the risks associated with mistakes in the implementation.

To describe precisely the approach, we define a taxonomy of the relevant classes of Verifiable Credentials that we need.

For the purpose of this discussion we use a diagram derived from the taxonomy [defined in EBSI](#):



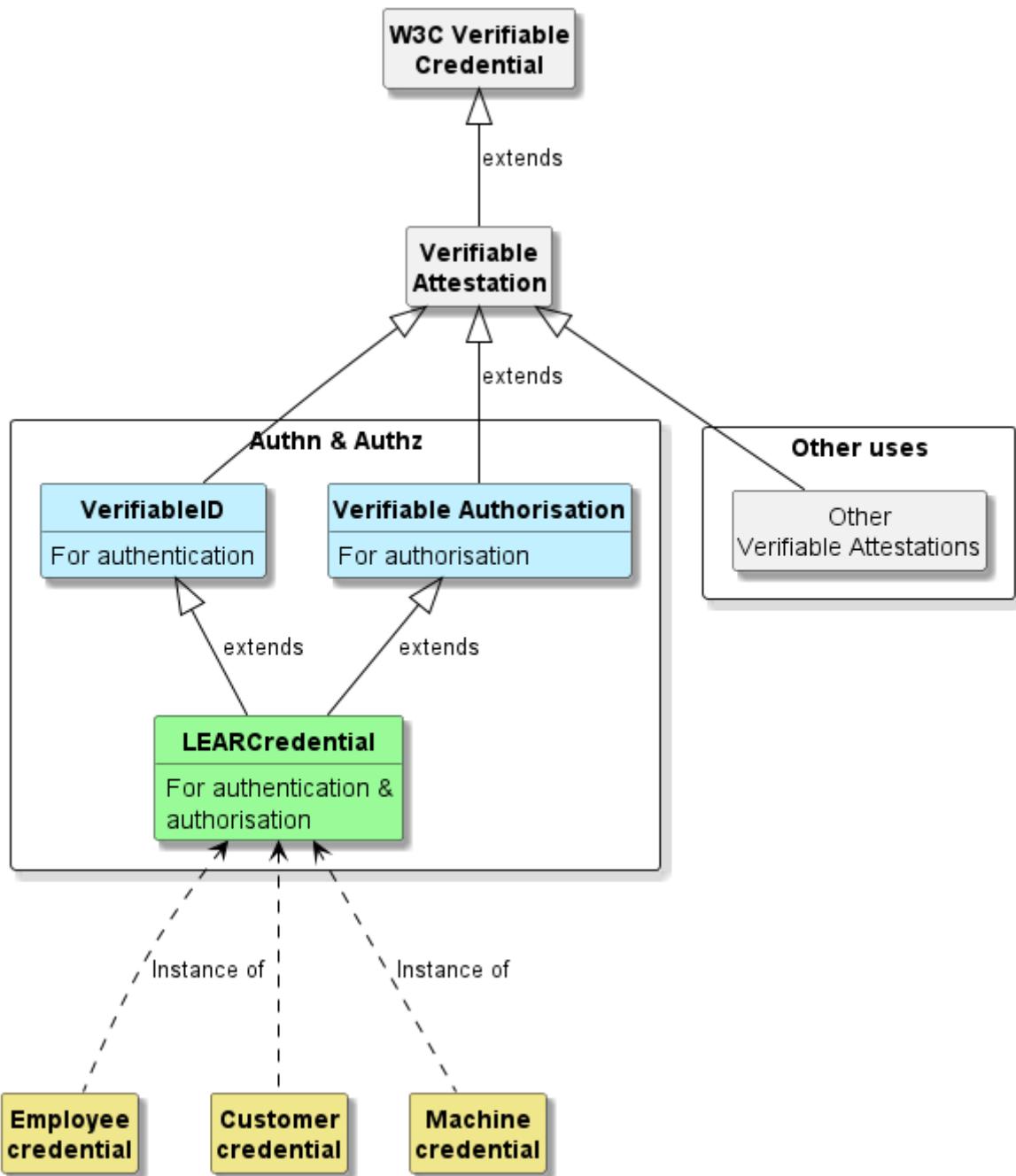


Figure 12 - Classes of credentials

The sections below refer to this diagram for each type of credential discussed.

3.3.1.3 Authentication requires a VerifiableID

Not all types of Verifiable Credentials can be used as a mechanism for online authentication, because the Relying Party (the entity receiving the Verifiable Credential in an authentication



process) needs to bind the identity of the user sending the credential to the claims attested about the subject inside the credential.

For example, in the case of a Diploma as a Verifiable Credential (a Verifiable Diploma), the credential is a Verifiable Attestation which indicates that the subject has certain skills or has achieved certain learning outcomes through formal or non-formal learning context. However, as in the case with normal Diplomas in paper or PDF format, this credential can be presented by anyone that holds the credential. In other words, the credential binds the identity of the subject with the claims inside the credential, but does not bind the identity of the holder of the credential with the identity of the subject of the credential, which requires a special mechanism in the issuance process.

Most Verifiable Credentials will be issued as Verifiable Attestations, acting as efficient machine-processable and verifiable equivalent to their analog counterparts. Their purpose is to attest some attributes about the subject of the credential and not act as a mechanism for online authentication.

Instead, a **VerifiableID** is a special form of a Verifiable Credential that a natural or legal person can use in an electronic identification process as evidence of whom he/she/it is (comparable with a passport, physical IDcard, driving licence, social security card, member-card...).

VerifiableIDs can be of different types and be issued by different entities with different levels of assurance (in eIDAS terminology). They are issued with the purpose of serving as authentication mechanisms in online processes. In the near future, VerifiableIDs of the highest level of assurance (LoA High) will be issued to citizens and businesses by their governments under eIDAS2.

We describe in this document a way to issue VerifiableID credentials and how a Relying Party can perform authentication by accepting that credential.

3.3.1.4 Access Control requires a Verifiable Authorisation

Once the user is authenticated, the system should make a decision to grant or reject an access request to a protected resource from an already authenticated subject, based on what the subject is authorised to access.

Let's take the example of when a Service Provider signs an agreement with a Service Consumer organisation (a legal person), and the Service Provider wants to allow access to some services to employees of the Consumer organisation under the conditions of the agreement.

In most cases, granting access is not an all-or-nothing decision. In general, the agreement between the Service Provider and Service Consumer specifies that only a subset of the employees of the Consumer organisation can access the services, and even in that subset not all employees have the same powers when accessing the services. For example, some employees have access to the administration of the service, some can perform some create/update/delete operations in a subset of the services, and some employees can only have read access to a subset of the services.



To allow for this granularity in an authentication and access control process, in general the entity willing to perform an action on a protected resource has to present (in a Verifiable Presentation) a VerifiableID and possibly one or more additional Verifiable Attestations.

At least one of the credentials should include claims identifying the employee (or any other subject) and a formal description of the concrete powers that have been delegated by the legal representative of the organisation, enabling the determination by the Service Provider whether the user is entitled to access a service and the operations that the user can perform on that service.

We define later a standard mechanism and format for a Verifiable Credential that enables this functionality in a simple, secure and with high degree of legal certainty compatible with eIDAS.

Such a credential is called a Verifiable Authorization, represented in the figure above.

3.3.1.5 Combining VerifiableID and Verifiable Authorisation in the LEARCredential

In many use cases, it is possible to simplify the authentication and access control by combining in a single credential both a VerifiableID (for authentication) and a Verifiable Authorisation (for access control).

The resulting credential is called LEARCredential and it is very useful when the holder/subject wants to use decentralised IAM mechanisms implemented by Relying Parties which are federated in a decentralised way using a common Trust Anchor Framework.

The concept of LEARCredential is described in the diagram above.

The user will authenticate using a special type of Verifiable Credential called **LEARCredential** (from **Legal Entity Authorised Representation**).

To achieve a high level of legal certainty under eIDAS, the LEARCredential is:

- signed or sealed with an eIDAS certificate which is either:
 - a **certificate for electronic seals** issued to a legal person by a Qualified Trust Service Provider, or
 - a **certificate for electronic signatures**, issued to a **legal representative of the legal person** by a Qualified Trust Service Provider.
- signed using the **JSON Advanced Electronic Signature** format described in [ETSI TS 119 182-1 V1.1.1 \(2021-03\) - Electronic Signatures and Infrastructures \(ESI\); JAdES digital signatures; Part 1: Building blocks and JAdES baseline signatures](#)

The LEARCredential includes claims identifying the user and a description of the concrete powers that have been delegated by the legal representative of the organisation.

By signing/sealing the credential with an eIDAS certificate, the legal representative attests that the powers described in the credential have been delegated to the user (maybe under some conditions, also described in the credential).



In this way, the LEARCredential has the same legal status as any other document in other formats (e.g., PDF) signed in an eIDAS-compliant way, but with the efficiency provided by the Verifiable Credential format.

In addition, the LEARCredential includes cryptographic material that allows the user to **employ the credential as an online authentication mechanism**, by proving that the holder of the credential is the same user identified in the subject of the credential.

Any Verifier that trusts the eIDAS Trust Framework will be able to verify that:

- **The user presenting the LEARCredential is the same as the one identified in the subject of the credential**
- **A legal representative of the organisation has attested that the user has the powers described in the credential**

This enables the user presenting the LEARCredential to start the process and to provide any additional required documentation, preferably as additional Verifiable Credentials to enable automatic verification of compliance with the process requirements (including Gaia-X credentials issued by the Compliance Service of Gaia-X).

Both types of eIDAS certificates mentioned above are an electronic attestation **that links electronic seal/signature validation data to a legal person and confirms the name of that person**. This way, the certificate, usually linked to the sealed/signed document, can be used to verify the identity of the creator of the seal/signature and whether the document has been sealed/signed using the corresponding private key.

Before issuing a certificate like the above, the Qualified Trust Service Provider (QTSP) performs validations against **Authentic Sources** to authenticate the identity of the organisation:

- The data concerning the business name or corporate name of the organisation.
- The data relating to the constitution and legal status of the subscriber.
- The data concerning the extent and validity of the powers of representation of the applicant.
- The data concerning the tax identification code of the organisation or equivalent code used in the country to whose legislation the subscriber is subject.

The person controlling the above certificates will appoint a legal entity representative (LEAR) by generating and signing a Verifiable Credential which:

- Includes identification data of an employee of the organisation
- Includes claims stating the delegation of specific powers to perform a set of specific processes on behalf of the organisation
- Includes a public key where the corresponding private key is controlled by the employee, enabling him to prove that he is the holder of the credential when it is being used in an authentication process like the ones used as examples in this document.



The LEARCredential uses the [did:elsi](#) method for the identifiers of legal persons involved in the process, to facilitate the DID resolution and linkage with the eIDAS certificates. The [did:elsi](#) method is specified in [DID ETSI Legal person Semantic Identifier Method Specification \(did:elsi\)](#).

The high-level view of an example process is described in the following diagram, when a COO (Chief Operating Officer) appoints an employee to perform some processes (this is an example instance of a LEARCredential):

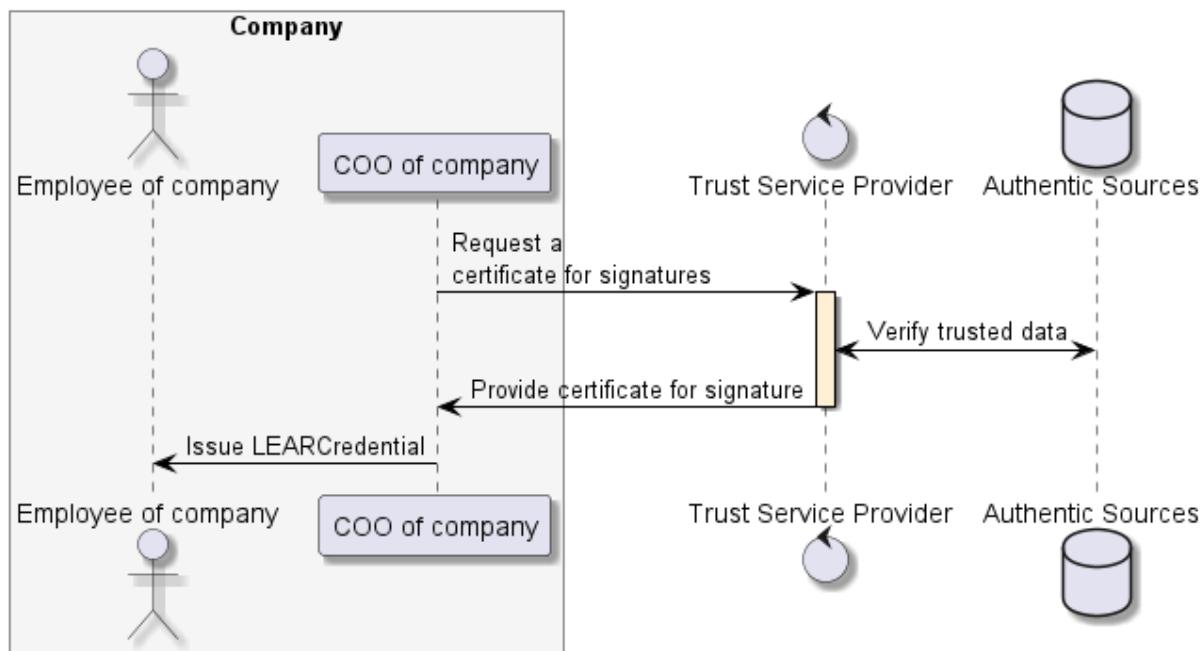


Figure 13 - High level view of issuance of LEARCredential

3.3.1.6 The LEARCredential for some identification processes

Any entity can issue VerifiableIDs, though the acceptance by Relying Parties can not be ensured because it is the Relying Party the only one deciding if it trusts on the issuer and its process for generating the VerifiableID.

DOME will make use of the future eIDAS2 VerifiableIDs when they are available and when they make sense in the context of DOME. But DOME defines some specific types of VerifiableIDs that are derived from eIDAS certificates for signatures/seals using the regulated standards for signatures, achieving a level of assurance that provides an acceptable level of legal certainty and reduced risk of repudiation.

An example is the LEARCredential described in a section above, which has the same level of legal certainty as any document signed with the same certificates (like a contract or an invoice).

Similarly, a Product provider (like GoodAir) who decides to use the DOME Trust and IAM framework or a compatible one for managing access to associated services by customers of

their products will typically define specific types of VerifiableIDs if the standard ones in DOME or the existing and widely accepted ones are not suitable for the provider. In general, if a provider can use an existing and already used VerifiableID then it will facilitate potential customers access to its product because issuers of the VerifiableID do not have to modify or adapt their existing issuing processes.

3.3.2 The LEARCredential through an example

In this section we describe in detail the LEARCredential and for illustrative purposes we use example attributes from a fictitious ecosystem whose participants are described in Appendix [Example scenario: Participants in the ecosystem](#).

In this example the LEARCredential will be generated using the eIDAS certificate of the COO (Chief Operation Officer) of the organisation that will be issuing the credential.

The credential will be generated with an application that the COO will use as Issuer of the LEARCredential and that allows the employee to receive the credential using his credential wallet, using [\[OpenID.VCI\]](#) to achieve compliance with the EUDI Wallet ARF.

This application enables the COO to attest the information required to create the LEARCredential, specifying the employee information and the specific type of LEARCredential. In general, there may be different instances of LEARCredentials for different purposes. One employee can have more than one LEARCredential, each having a different delegation of powers for different environments.

The specific detailed use case here is the issuance of a LEARCredential to an employee to enable that employee to perform the onboarding process in an ecosystem. But exactly the same process with different attested attributes can be used by any organisation to issue credentials that can be accepted by other organisations for authentication and access control to any protected resources that they manage.

The essential components of a LEARCredential are the following:

- **Claims identifying the employee**
- **DID of the employee to enable authentication**
- **Claims identifying the legal representative**
- **The roles and duties of the LEAR**
- **Advanced/Qualified signature of the credential**

These concepts are elaborated in the following sections.

3.3.2.1 Claims identifying the employee

These are claims identifying the subject of the credential, the person who will act as LEAR. Each ecosystem can define their own depending on their specific requirements.



In addition, each organisation can specify their own for enabling access to its products/services. However, it is recommended to use a set of claims that are agreed upon by all participants in the ecosystem unless there are specific requirements for not doing so. This includes not only the amount of claims but also their syntax and semantics.

For our example, we use the same that are used for accessing the European Commission portal. The claims in the credential are the digital equivalent of their analog counterparts, displayed here for illustration.



EU Funding & Tenders: LEAR appointment letter: V6.0 – 01.09.2022

BG CS DA DE EL EN ES ET FI FR GA HR HU IT LT LV MT NL PL PT RO SK SL SV

LEAR APPOINTMENT LETTER

(This document will be automatically generated by the Participant Register once all the information required for the LEAR appointment will have been filled in. You should print it, have it signed by the legal representative and the LEAR and then upload it in the Participant Register with the supporting documents. Originals should be kept on file for controls. If you would like to consult other language versions, please refer to templates & forms section of the [Portal Reference Documents page](#).)

Subject: **PIC:**
Legal entity name:

I, Mr/Ms/Mrs/Miss, in my capacity as and authorised to legally represent my organisation, have appointed as our legal entity appointed representative (LEAR):

First name:
Last name:
Title: Mr/Ms/Mrs/Miss
Gender:
Postal address (street, postcode, city and country):
e-mail:
Telephone: +(...)
Fax: +(...)
Mobile Phone¹: +(...)

¹ The activation of the LEAR account requires the log in with a PIN code. If you provide a mobile phone number, this PIN code can be sent by SMS. Otherwise we have to send it by post. The number will be used exclusively for sending the PIN code.

Figure 14 - [Figure LEAR subject identification data](#).



From the above form we can derive the following claims using the example data:

```
{
  "title": "Mr.",
  "first_name": "John",
  "last_name": "Doe",
  "gender": "M",
  "postal_address": "",
  "email": "johndoe@goodair.com",
  "telephone": "",
  "fax": "",
  "mobile_phone": "+34787426623"
}
```

3.3.2.2 DID of the employee

In this example we use the `did:key` method for the DID of the employee, which provides a very high level of privacy and given the way the LEARCredential is generated it supports the verification of the chain of responsibility with a proper level of guarantee.

If the highest levels of assurance and legal compliance is desired and the employee has an eIDAS certificate for signatures (in this case a personal one, not one like the COO), we could use the `did:elsi` method for identification of the employee. However, the organisation (GoodAir in our example) should make sure that the employee is aware of and agrees to the possible privacy implications of doing so, given the personal details leaked from the eIDAS certificate.

Those personal details are exactly the same as if the employee signs any document with a digital certificate, but care should be taken by the organisation because in this case the signature would be done "on behalf of" his employer and not as an individual personal action.

Even though this is not unique to the `did:elsi` method, this also implies that the onboarding service has to handle those personal details in the same way as if it would be accepting any other document signed with a certificate for signatures, and ensure compliance with GDPR. This is "business as usual" when using digital signatures, but we want to make sure that this is taken care of when implementing the authentication and access control mechanisms described here.

In this example we assume the usage of the `did:key` method for the employee to protect his privacy as much as possible.

This DID for the employee is an additional claim to the ones presented above, using the `id` field in the `credentialSubject` object.

Using this DID method has an additional advantage: the DID identifier corresponds to a keypair that is generated during the LEARCredential issuance process, where the private key is generated by the wallet of the employee and it is always under his control.



This private key controlled by the employee can be used to sign challenges from Relying Parties that receive the credential to prove that the person sending the credential is the same person that is identified in the `credentialSubject` object of the LEARCredential.

It is this property that makes this credential a VerifiableID that can be used for online authentication.

An example DID for the employee could be:

```
{  
  "id": "did:key:z6MkhaXgBZDvotDkL5257faiztiGiC2QtKLGbnnEGta2doK"  
}
```

In this example, the signatures performed with the private key can not be JAdES-compliant ([ETSI-JADES]), but if the LEARCredential is attached to any other credential that is signed with this private key, then they can be traced up to the eIDAS certificate of the COO and so the chain of responsibility can be determined..

With the DID for the employee, the set of claims identifying him would be then:

```
{  
  "id": "did:key:z6MkhaXgBZDvotDkL5257faiztiGiC2QtKLGbnnEGta2doK",  
  "title": "Mr.",  
  "first_name": "John",  
  "last_name": "Doe",  
  "gender": "M",  
  "postal_address": "",  
  "email": "johndoe@goodair.com",  
  "telephone": "",  
  "fax": "",  
  "mobile_phone": "+34787426623"  
}
```

3.3.2.3 legalRepresentative

This section identifies the natural person (the COO) who is a legal representative of the legal person (GoodAir) and that is nominating the employee identified in the credential.



```
"legalRepresentative": {  
    "cn": "56565656V Jesus Ruiz",  
    "serialNumber": "56565656V",  
    "organizationIdentifier": "VATES-12345678",  
    "o": "GoodAir",  
    "c": "ES"  
}
```

NOTE: Attributes for natural and legal persons

The attributes for natural persons and legal persons are derived from the [eIDAS SAML Attribute Profile \(eIDAS Technical Sub-group, 22 June 2015\)](#).

All attributes for the eIDAS minimum data sets can be derived from the [ISA Core Vocabulary](#) and <https://joinup.ec.europa.eu/collection/semic-support-centre/specifications>.

In the case of natural persons refer to the [Core Person Vocabulary](#) and in the case of legal persons refer to definitions for [Core Business Vocabulary](#).

3.3.2.4 rolesAndDuties of the LEAR

The LEARCredential should include a formal description with the roles and duties of the LEAR. This is the digital equivalent to the analog description in our example, which uses natural language:



EU Funding & Tenders: LEAR appointment letter: V6.0 – 01.09.2022

BG CS DA DE EL EN ES ET FI FR GA HR HU IT LT LV MT NL PL PT RO SK SL SV

ROLES AND DUTIES OF LEARS

1. What is a LEAR?

LEAR stands for **legal entity appointed representative**.

For organisations (i.e. not individuals), this is a person formally appointed by the legal representative of the organisation to perform certain tasks on behalf of their organisation, as part of its participation in EU funded grants, procurements and prizes that are managed via the [EU Funding & Tenders Portal](#) — the EU's dedicated website for funding and tenders.

Individuals automatically have the role of LEAR.

2. What can a LEAR do?

As a LEAR you can:

- **view** your organisation's legal and financial data in the Participant Register
- ask to validate **updates of** this information where necessary
- monitor whether or not this information is **validated**, and when
- monitor all uses made of your organisation's **participant identification code** (PIC).

3. What must you do?

As a LEAR you have certain formal obligations:

- **provide** up-to-date legal and financial data (including — on request — supporting documents) on your organisation.
- **Maintain and update** this data (*i.e. enabling it to be used for contracting and other transactions between your organisation and the EU*). This means you must **regularly check** that the data is correct and immediately request changes.
- enter and update the names of the colleagues authorised to act as **legal representatives and signatories** for your organisation. These are people who are able to commit your organisation legally by signing grant agreements or contracts and authorising amendments to them.

You must also **revoke** this assignment for any colleague who no longer has these powers.

- enter and update the names of any colleagues authorised to **sign financial statements or invoices** on behalf of your organisation.

You must also **revoke** this assignment for any colleague who no longer has this authorisation.

Figure 15 - [Figure LEAR roles and duties](#).



We do not yet have defined the actual mechanism that will be used for describing formally the roles and duties of the LEAR. The way we specify them below is just an example. There is work ongoing to define the detailed mechanism formally (e.g. with ODRL), and it is expected to change. This example is intended only to describe the concepts in detail.

The `rolesAndDuties` object points to an externally hosted object with the roles and duties of the LEAR. This external object can be either a machine-interpretable definition of the roles and duties in the credential, or just an external definition of the roles and duties in natural language. The ideal approach is the first option, expressing the semantics with a proper machine-readable language, because this will allow automatic access control at the granularity of the individual sentences of that expression language. The `rolesAndDuties` object can also have the definition embedded into it, instead of having a pointer to an external object.

A simplistic implementation of the object inside the credential could be:

```
"rolesAndDuties": [
  {
    "type": "LEARCredential",
    "id": "https://dome-marketplace.eu/lear/v1/6484994n4r9e990494"
  }
]
```

Where the last part of the url can correspond to the hash of the external linked document to ensure that any modification or tampering can be detected. The contents of that object at that url could be:

```
[
  {
    "id": "https://dome-marketplace.eu/lear/v1/6484994n4r9e990494",
    "target": "https://bae.dome-marketplace.eu/",
    "roleNames": ["seller", "customer"]
  }
]
```

In the above example, we have specified that the roles object referenced in the credential with `id: https://dome-marketplace.eu//lear/v1/6484994n4r9e990494` is granting the employee of GoodAir (in our example) to access services in the DOME marketplace (specified with the field `target`) with the roles `seller` and `customer` (which are the roles of the [FIWARE BAE component](#)).



If `target` is omitted, the roles would apply to any target organisation. If we need to specify more than one target organisation, the array should include as many objects as required.

3.3.2.5 Assembling the pieces together

With the above values for the example, the complete LEARCredential would become something like this:

```
{
  "@context": [
    "https://www.w3.org/ns/credentials/v2",
    "https://dome-marketplace.eu//2022/credentials/learcredential/v1"
  ],
  "id": "urn:did:elsi:25159389-8dd17b796ac0",
  "type": ["VerifiableCredential", "LEARCredential"],
  "issuer": {
    "id": "did:elsi:VATES-12345678"
  },
  "issuanceDate": "2022-03-22T14:00:00Z",
  "validFrom": "2022-03-22T14:00:00Z",
  "expirationDate": "2023-03-22T14:00:00Z",
  "credentialSubject": {
    "id": "did:key:z6MkhaXgBZDvotDkL5257faiztiGiC2QtKLGpbnnEGta2dok",
    "title": "Mr.",
    "first_name": "John",
    "last_name": "Doe",
    "gender": "M",
    "postal_address": "",
    "email": "johndoe@goodair.com",
    "telephone": "",
    "fax": "",
    "mobile_phone": "+34787426623",
    "legalRepresentative": {
      "cn": "56565656V Jesus Ruiz",
      "serialNumber": "56565656V",
      "organizationIdentifier": "VATES-12345678",
      "o": "GoodAir",
      "c": "ES"
    },
    "rolesAndDuties": [
      {
        "type": "LEARCredential",
        "id": "https://dome-marketplace.eu//lear/v1/6484994n4r9e990494"
      }
    ]
  }
}
```



3.3.3 Issuing the LEARCredential

3.3.3.1 Overview

In this example we use what we call a *profile* of the [OpenID.VCI] protocol. The standard is very flexible, and we restrict the different options available in the standard and implement a set of the options with given values that are adequate for our use case, without impacting flexibility in practice.

Here we describe the flows and data structures comprehensively, including relevant excerpts of the OpenID specification in order to freeze a concrete specification for DOME development and enhance interoperability among components developed by different parties. New versions of this document will be made in the future and will evolve gradually to keep it in sync with the latest versions of the OpenID specifications, ensuring that components in DOME are updated in sync and are interoperable among them.

The following figure describes the main components that interact in the issuance of a credential in this profile.

The description of the issuance process is general enough to be used for many types of credentials, but the text includes notes describing the concrete application of the process to the case of the LEARCredential.

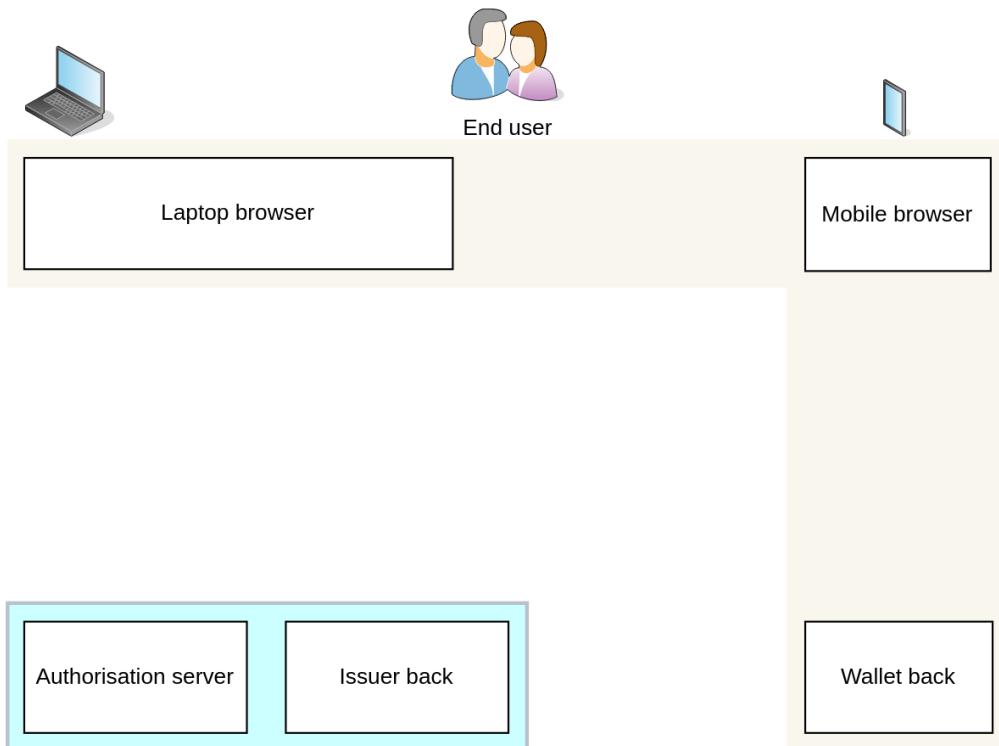


Figure 16 - Figure Overview of participants.

End user

This profile is valid for both natural and juridical persons, but because we are focusing on the issuance of the LEARCredential, in the detailed examples below we assume a natural person as the user.

Wallet

The wallet is assumed to be a Web application with a wallet backend, maybe implemented as a PWA so it has some offline capabilities and can be installed in the device, providing a user experience similar to a native application. Private key management and most sensitive operations are performed in a backend server, operated by an entity trusted by the end user. Other profiles can support native and completely offline PWA mobile applications, for end users.

This type of wallet supports natural persons, juridical persons and natural persons who are legal entity representatives of juridical persons. For juridical persons the wallet is usually called an **enterprise wallet** but we will use here just the term **wallet** unless the distinction is required.

In this profile we assume that the wallet is not previously registered with the Issuer and that the wallet does not expose public endpoints that are called by the Issuer, even if the wallet has a backend server that could implement those endpoints. That makes the wallet implementations in this profile to be very similar in interactions to a full mobile implementation, making migration to a full mobile implementation easier.

In other words, from the point of view of the Issuer, the wallet in this profile is almost indistinguishable from a full mobile wallet.

User Laptop

For clarity of exposition, we assume in this profile that the End User starts the interactions with the Issuer with an internet browser (user agent) in her laptop. However, there is nothing in the interactions which limits those interactions to a laptop form factor and the End User can interact with any internet browser in any device (mobile, tablet, kiosk).

Issuer

In this profile we assume that the Issuer is composed of two components:

- **Authorization server:** the backend component implementing the existing authentication/authorization functionalities for the Issuer entity.
- **Issuer backend:** the main server implementing the business logic as a web application and additional backend APIs required for issuance of credentials.

The Issuer backend and the Authorization server could be implemented as a single component in an actual deployment, but we assume here that they are separated to make the profile more general, especially for big entities and also when using Trust Service Providers for cloud signature and credential issuance, for example.

Authentication of End User and previous Issuer-End User relationship



We assume that the Issuer and End User have a previous relationship and that the Issuer has performed the KYC required by regulation and needed to be able to issue Verifiable Credentials attesting some attributes about the End User. We assume that there is an existing trusted authentication mechanism (not necessarily related to Verifiable Credentials) that the End User employs to access protected resources from the Issuer. For example, the user is an employee or a customer of the Issuer, or the Issuer is a Local Administration and the End User is a citizen living in that city.

3.3.3.2 Authentication

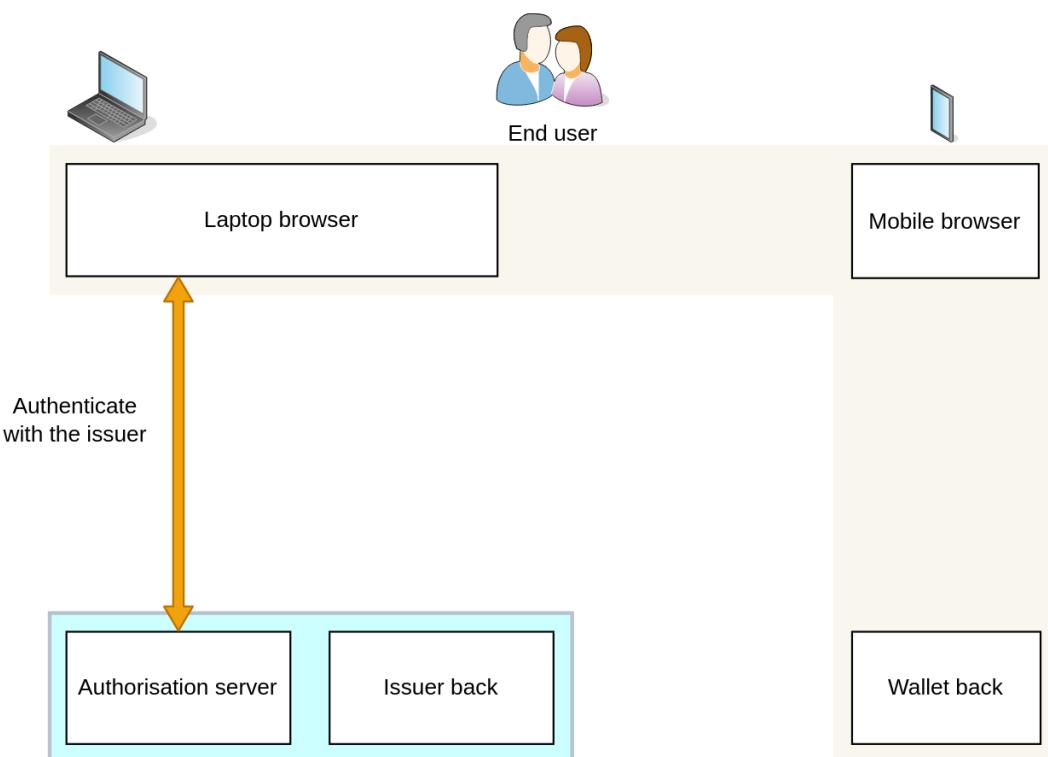


Figure 17 - Figure_Issuance authentication.

Before requesting a new credential, the End User has to authenticate with the Issuer with whatever mechanism is already implemented by the Issuer. This profile does not require that it is based on OIDC, Verifiable Credentials or any other specific mechanism.

The level of assurance (LoA) of this authentication mechanism is one of the factors that will determine the confidence that the Verifiers can have on the credentials received by them from a given Issuer.

NOTE: LEARCredential

In the case of the LEARCredential, the End User (John Doe) is an employee of GoodAir and in order to receive the credential John first has to authenticate into the company systems using whatever mechanism GoodAir uses for employee authentication.

Being a modern company, GoodAir uses Verifiable Credentials IAM but this is not a requirement for the issuance of the LEARCredential.

3.3.3.3 Credential Offer

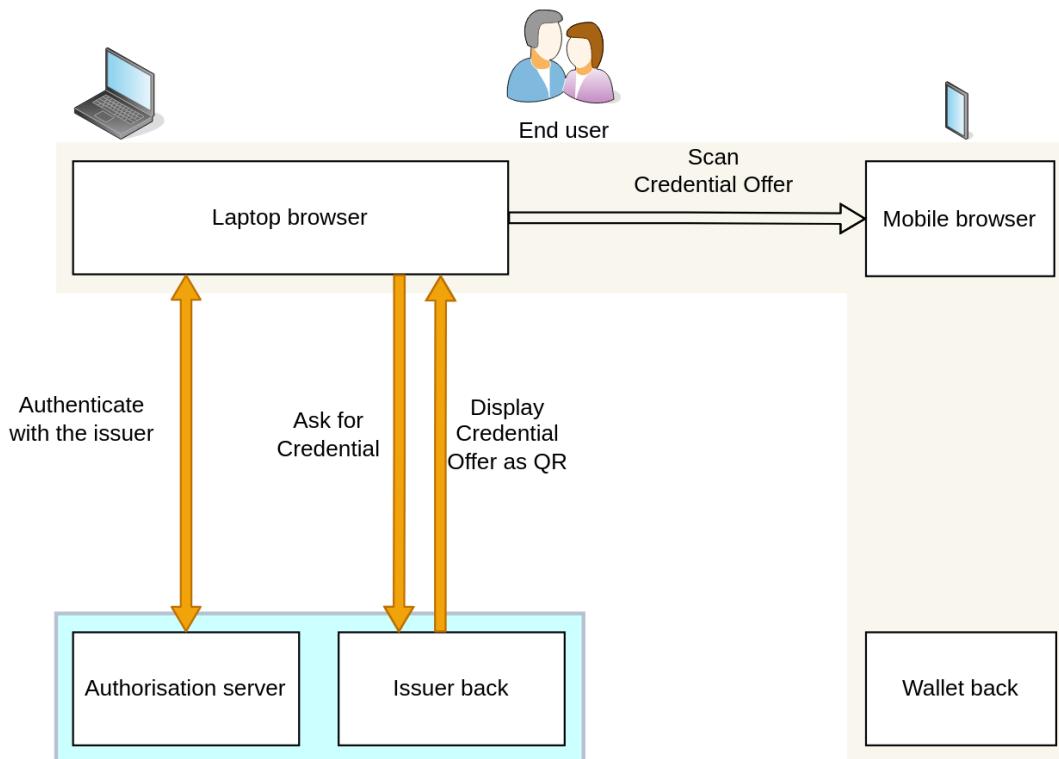


Figure 18 - [Figure](#) Credential offer.

In this profile the wallet does not have to implement the Credential Offer Endpoint described in section 4 of [[OpenID.VCI](#)].



Instead, the Credential Issuer renders a QR code containing a reference to the Credential Offer that can be scanned by the End-User using a Wallet, as described in section 4.1 of [OpenID.VCI].

According to the spec, the Credential Offer object is a JSON object containing the Credential Offer parameters and can be sent by value or by reference. To avoid problems with the size of the QR travelling in the URL, this profile requires that the QR contains the `credential_offer_uri`, which is a URL using the `https` scheme referencing a resource containing a JSON object with the Credential Offer parameters. The `credential_offer_uri` endpoint should be implemented by the Issuer backend.

3.3.3.3.1 Credential Offer Parameters

This profile restricts the options available in section 4.1.1 of [OpenID.VCI]. The profile defines a Credential Offer object containing the following parameters:

- `credential_issuer`: *REQUIRED*. The URL of the Credential Issuer that will be used by the Wallet to obtain one or more Credentials.
- `credentials`: *REQUIRED*. A JSON array, where every entry is a JSON string. To achieve interoperability faster, this profile defines a global Trusted Credential Schemas List where well-known credential schemas are defined, in addition to the individual credentials that each Issuer can define themselves. The string value *MUST* be one of the id values in one of the objects in the `credentials_supported` metadata parameter of the Trusted Credential Schemas List (described later), or one of the id values in one of the objects in the `credentials_supported` Credential Issuer metadata parameter provided by the Credential Issuer. When processing, the Wallet *MUST* resolve this string value to the respective object. The credentials defined in the global Trusted Credential Schema List have precedence over the ones defined by the Credential Issuer.

NOTE: LEARCredential

The only credential being offered in our case is the LEARCredential, so this is the credential schema that should be specified here. The LEARCredential is a credential known globally to the DOME ecosystem, so its schema should be published and be available in the Trusted Credential Schemas List.

- `grants`: *REQUIRED*. A JSON object indicating to the Wallet the Grant Type `pre-authorized_code`. This grant is represented by a key and an object, where the key is `urn:ietf:params:oauth:grant-type:pre-authorized_code`. In this profile the credential issuance flow requires initial authentication of the End User by the Credential Issuer, so the Pre-Authorized Code Flow achieves a good level of security and we do not need the more general Authorization Code Flow.



In other scenarios like when the wallet is a native mobile application and the user interacts with the Issuer exclusively with the mobile (without the laptop), then the Authorization Code Flow has to be used. This can be described in detail in a different profile.

The grant object contains the following values:

- **pre-authorized_code**: *REQUIRED*. The code representing the Credential Issuer's authorization for the Wallet to obtain Credentials of a certain type. This code *MUST* be short lived and single-use. This parameter value *MUST* be included in the subsequent Token Request with the Pre-Authorized Code Flow.
- **user_pin_required**: *REQUIRED*. The [OpenID.VCI] standard says it is *RECOMMENDED*, but this profile specifies the user pin to achieve a greater level of security. This field is a boolean value specifying whether the Credential Issuer expects presentation of a user PIN along with the Token Request in a Pre-Authorized Code Flow. Default is false. This PIN is intended to bind the Pre-Authorized Code to a certain transaction in order to prevent replay of this code by an attacker that, for example, scanned the QR code while standing behind the legit user. It is *RECOMMENDED* to send a PIN via a separate channel. The PIN value *MUST* be sent in the **user_pin** parameter with the respective Token Request.

The following non-normative example shows a Credential Offer object where the Credential Issuer offers the issuance of one Credential ("LEARCredential"):

```
{
  "credential_issuer": "https://www.goodair.com",
  "credentials": [
    "LEARCredential"
  ],
  "grants": {
    "urn:ietf:params:oauth:grant-type:pre-authorized_code": {
      "pre-authorized_code": "asju68jgtyk9ikkew",
      "user_pin_required": true
    }
  }
}
```

3.3.3.3.2 Contents of the QR code

Below is a non-normative example of the Credential Offer displayed by the Credential Issuer as a QR code when the Credential Offer is passed by reference, as required in this profile:

```
https://www.goodair.com/credential-offer?
credential_offer_uri=https%3A%2F%2Fserver%2Eexample%2Ecom%2Fcredential-
offer%2F5j349k3e3n23j
```



Which in plain text would be:

```
https://www.goodair.com/credential-offer?  
credential_offer_uri=https://www.goodair.com/credential-offer/5j349k3e3n23j
```

To increase security, the Issuer *MUST* make sure that every Credential Offer URI is unique for all credential offers created. This is the purpose of the nonce (**5j349k3e3n23j**) at the end of the url in the example. Issuers can implement whatever mechanism they wish, as far as it is transparent to the wallet.

3.3.3.4 Credential Issuer Metadata

The Wallet backend retrieves the Credential Issuer's configuration using the Credential Issuer Identifier that was received in the Credential Offer.

A Credential Issuer is identified in this context by a case sensitive URL using the https scheme that contains scheme, host and, optionally, port number and path components, but no query or fragment components. No DID is used in this context.

Credential Issuers *MUST* make a JSON document available at the path formed by concatenating the string **/.well-known/openid-credential-issuer** to the Credential Issuer Identifier. If the Credential Issuer value contains a path component, any terminating / *MUST* be removed before appending **/.well-known/openid-credential-issuer**.

openid-credential-issuer *MUST* point to a JSON document compliant with this specification and *MUST* be returned using the **application/json** content type.

The retrieval of the Credential Issuer configuration is illustrated below.



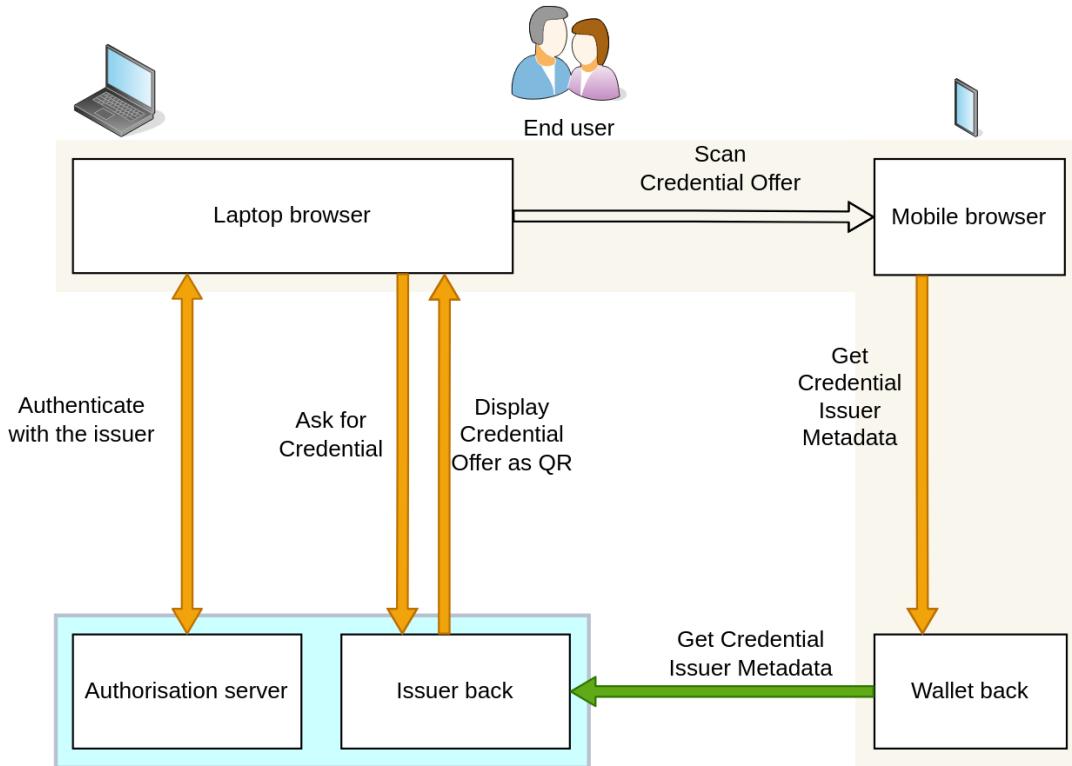


Figure 19 - [Figure_Issuer metadata](#).

3.3.3.4.1 Credential Issuer Metadata Parameters

The object contained in `openid-credential-issuer` contains the following:

- `credential_issuer`: *REQUIRED*. The Credential Issuer's identifier.
- `credential_endpoint`: *REQUIRED*. URL of the Credential Issuer's Credential Endpoint. This URL *MUST* use the https scheme and *MAY* contain port, path and query parameter components.
- `credentials_supported`: *REQUIRED*. A JSON array containing a list of JSON objects, each of them representing metadata about a separate credential type that the Credential Issuer can issue. The JSON objects in the array *MUST* conform to the structure of the Section XXXX.

TODO: define a global directory of credentials supported to eliminate requirement for each individual Issuer to publish its own list.

This profile does not make use of the following parameters:

- `authorization_server` parameter, because it uses the `pre-authorized_code` Grant type.
- `batch_credential_endpoint` parameter. It indicates that the Credential Issuer does not support the Batch Credential Endpoint.
- `display` parameter.

3.3.3.5 OAuth 2.0 Authorization Server Metadata

This specification also defines a new OAuth 2.0 Authorization Server metadata [RFC8414] parameter to publish whether the AS that the Credential Issuer relies on for authorization, supports anonymous Token Requests with the Pre-authorized Grant Type. It is defined as follows:

- `pre-authorized_grant_anonymous_access_supported`: OPTIONAL. A JSON Boolean indicating whether the issuer accepts a Token Request with a Pre-Authorized Code but without a client id. The default is false.

3.3.3.6 Access Token

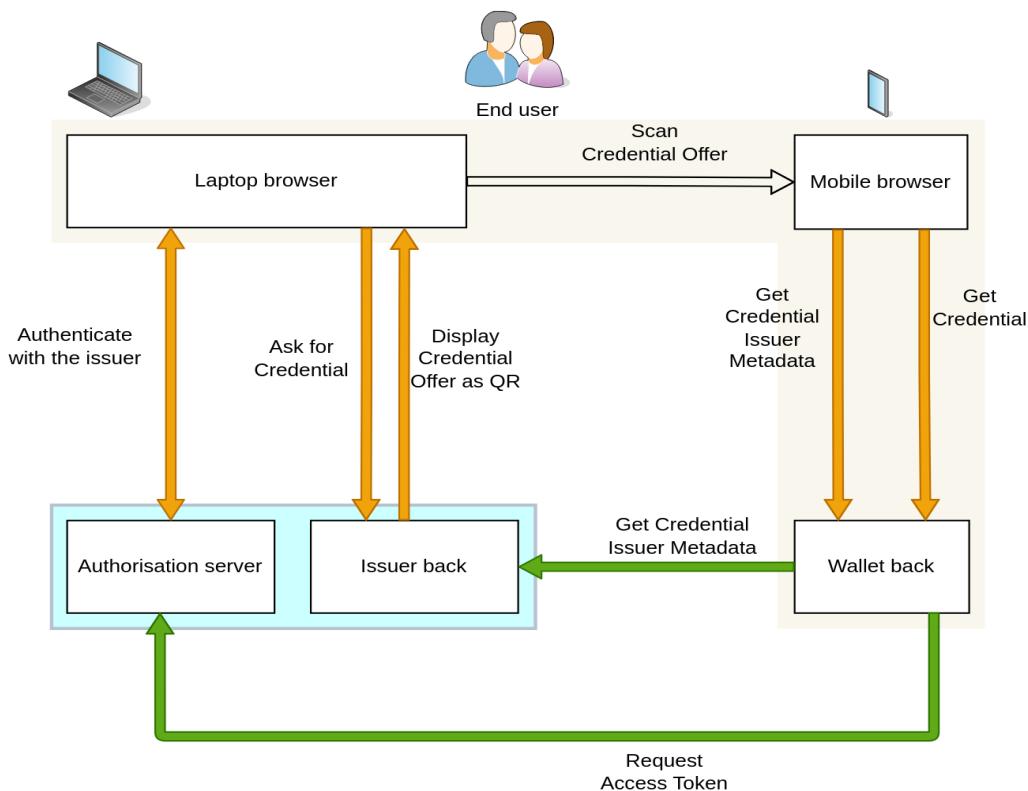


Figure 20 - [Figure Access Token](#).

The Wallet invokes the Token Endpoint implemented by the Authorization Server, which issues an Access Token and, optionally, a Refresh Token in exchange for the Pre-authorized Code that the wallet obtained in the Credential Offer.

3.3.3.6.1 Token Request

After the wallet receives the Credential Issuer Metadata, a Token Request is made as defined in Section 4.1.3 of [RFC6749].

The following are the extension parameters to the Token Request used in a Pre-Authorized Code Flow as used in this profile:

- **pre-authorized_code**: REQUIRED. The code representing the authorization to obtain Credentials of a certain type.
- **user_pin**: OPTIONAL. String value containing a user PIN. This value *MUST* be present if user_pin_required was set to true in the Credential Offer. The string value *MUST* consist of a maximum of 8 numeric characters (the numbers 0 - 9).

In this profile the Wallet does not have to authenticate when using the Token Endpoint, because we are using the Pre-Authorized Code Grant Type, given the level of trust between the Issuer and the End User and that authentication was already performed at the beginning of the flow.

Below is a non-normative example of a Token Request:

```
POST /token HTTP/1.1
Host: server.example.com
Content-Type: application/x-www-form-urlencoded

grant_type=urn%3Aietf%3Aparams%3Aoauth%3Agrant-type%3Apre-authorized_code
&pre-authorized_code=Spx10BeZQQYbYS6WxSbIA
&user_pin=493536
```

3.3.3.6.2 Successful Token Response

Token Responses are made as defined in [RFC6749].

In addition to the response parameters defined in [RFC6749], the Authorization Server returns the following parameters:

- **c_nonce**: REQUIRED. JSON string containing a nonce to be used to create a proof of possession of key material when requesting a Credential. The Wallet *MUST* use this nonce value for its subsequent credential requests until the Credential Issuer provides a fresh nonce.
- **c_nonce_expires_in**: REQUIRED. JSON integer denoting the lifetime in seconds of the c_nonce.

Below is a non-normative example of a Token Response:



```
HTTP/1.1 200 OK
```

```
Content-Type: application/json
```

```
Cache-Control: no-store
```

```
{
```

```
    "access_token": "eyJhbGciOiJSUzI1NiIsInR5cCI6Ikp..shQ",
    "token_type": "bearer",
    "expires_in": 86400,
    "c_nonce": "tZignsnFbp",
    "c_nonce_expires_in": 86400
}
```

```
}
```

3.3.3.6.3 Token Error Response

If the Token Request is invalid or unauthorised, the Authorization Server constructs the error response as defined in section 6.3 of [OpenID.VCI].

3.3.3.7 Request and receive Credential

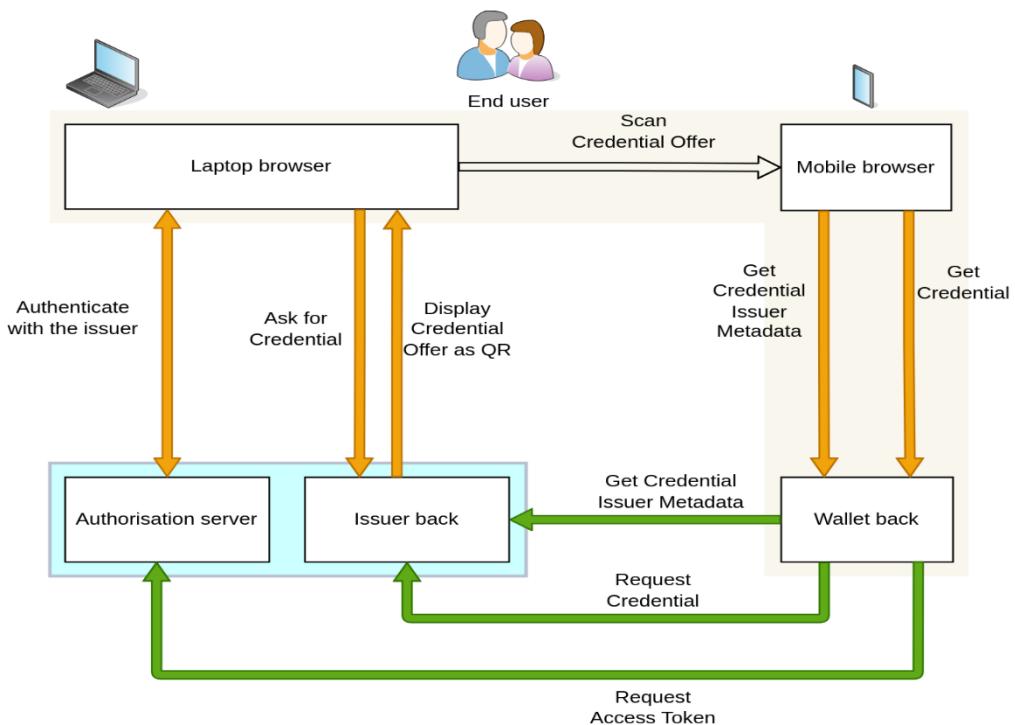


Figure 21 - Figure Request and receive Credential.



The Wallet backend invokes the Credential Endpoint, which issues a Credential as approved by the End-User upon presentation of a valid Access Token representing this approval.

Communication with the Credential Endpoint *MUST* utilise TLS.

The client can request issuance of a Credential of a certain type multiple times, e.g., to associate the Credential with different public keys/Decentralised Identifiers (DIDs) or to refresh a certain Credential.

If the Access Token is valid for requesting issuance of multiple Credentials, it is at the client's discretion to decide the order in which to request issuance of multiple Credentials requested in the Authorization Request.

3.3.3.7.1 Binding the Issued Credential to the identifier of the End-User possessing that Credential

The Issued Credential *MUST* be cryptographically bound to the identifier of the End-User who possesses the Credential. Cryptographic binding allows the Verifier to verify during the presentation of a Credential that the End-User presenting a Credential is the same End-User to whom that Credential was issued.

The Wallet has to provide proof of control alongside key material using the mechanism described below.

3.3.3.7.2 Credential Request

The Wallet backend makes a Credential Request to the Credential Endpoint by sending the following parameters in the entity-body of an HTTP POST request using the [application/json](#) media type.

- **format:** *REQUIRED*. This profile uses the Credential format identifier [jwt_vc_json](#).
- **proof:** *OPTIONAL*. JSON object containing proof of possession of the key material the issued Credential shall be bound to. The specification envisions use of different types of proofs for different cryptographic schemes. The proof object *MUST* contain a [proof_type](#) claim of type JSON string denoting the concrete proof type. This type determines the further claims in the proof object and its respective processing rules. Proof types are defined in Section [2.3.7.2.1 Proof Type](#).

The proof element *MUST* incorporate a [c_nonce](#) value generated by the Credential Issuer and the Credential Issuer Identifier (audience) to allow the Credential Issuer to detect replay. The way that data is incorporated depends on the proof type. In a JWT, for example, the [c_nonce](#) is conveyed in the [nonce](#) claim whereas the audience is conveyed in the [aud](#) claim. In a Linked Data proof, for example, the [c_nonce](#) is included as the challenge element in the proof object and the Credential Issuer (the intended audience) is included as the domain element.

3.3.3.7.2.1 PROOF TYPE

This specification defines only one value for [proof_type](#):



jwt: objects of this type contain a single jwt element with a JWS [RFC7515] as proof of possession. The JWT *MUST* contain the following elements:

- in the JOSE Header,
 - **typ**: *REQUIRED*. *MUST* be `openid4vci-proof+jwt`, which explicitly types the proof JWT as recommended in Section 3.11 of [RFC8725].
 - **alg**: *REQUIRED*. A digital signature algorithm identifier such as per IANA "JSON Web Signature and Encryption Algorithms" registry. *MUST NOT* be none or an identifier for a symmetric algorithm (MAC).
 - **kid**: *REQUIRED*. JOSE Header containing the key ID. The Credential will be bound to a DID, so the kid refers to a DID URL which identifies a particular key in the DID Document that the Credential will be bound to.
- in the JWT body,
 - **aud**: *REQUIRED* (string). The value of this claim *MUST* be the Credential Issuer URL of the Credential Issuer.
 - **iat**: *REQUIRED* (number). The value of this claim *MUST* be the time at which the proof was issued using the syntax defined in [RFC7519].
 - **nonce**: *REQUIRED* (string). The value type of this claim *MUST* be a string, where the value is the `c_nonce` provided by the Credential Issuer.

The Credential Issuer *MUST* validate that the proof is actually signed by a key identified in the JOSE Header.

Below is a non-normative example of a proof parameter (dots in the middle of jwt for display purposes only), for the example of issuing a LEARCredential:

```
{
  "proof_type": "jwt",
  "jwt": "eyJraWQiOiJkaWQ6ZXhhb...aZKPxgi hac0aW9EkL1nOzM"
}
```

where the JWT looks like this:

```
{
  "typ": "openid4vci-proof+jwt",
  "alg": "ES256",
  "kid": "did:key:z6MkhaXgBZDvotDkL5257faiztiGiC2QtKLGpbnnEGta2doK"
}

{
  "iss": "s6BhdRkqt3",
  "aud": "https://www.goodair.com",
  "iat": 1659145924,
  "nonce": "tZignsnFbp"
}
```



In the example of a LEARCredential, the wallet generates a pair of public/private keys and a **did:key** identifier which is univocally related to the public key. This is the reason why the **kid** field above is exactly the DID identifier under this DID method. The **did:key** method is very simple and achieves a very high degree of privacy, allowing the creation of many different identifiers which can be one-use only if so desired.

The **did:key** method is perfect for the requirements of our usage of the LEARCredential. Any other suitable DID method can be used if it is required, but this is out of scope for this profile.

3.3.3.7.3 Credential Response

This profile restricts Credential Response to be Synchronous and Deferred response is not used. The Credential Issuer *MUST* be able to immediately issue a requested Credential and send it to the Client.

The following claims are used in the Credential Response:

- **format**: *REQUIRED*. JSON string denoting the format of the issued Credential. This profile uses the format identifier **jwt_vc_json**.
- **credential**: *REQUIRED*. Contains issued Credential. *MUST* be a JSON string.
- **c_nonce**: *OPTIONAL*. JSON string containing a nonce to be used to create a proof of possession of key material when requesting a Credential. When received, the Wallet *MUST* use this nonce value for its subsequent credential requests until the Credential Issuer provides a fresh nonce.
- **c_nonce_expires_in**: *OPTIONAL*. JSON integer denoting the lifetime in seconds of the c_nonce.

Below is a non-normative example of a Credential Response:

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

{
  "format": "jwt_vc_json",
  "credential" : "LUpixVCWJk0e0t4CXQe1NXK....WZwmhm90Qp6YxX0a2L",
  "c_nonce": "fGFF7UkhLa",
  "c_nonce_expires_in": 86400
}
```

3.3.3.7.4 Credential Error Response

When the Credential Request is invalid or unauthorised, the Credential Issuer constructs the error response as defined in section 7.3.1 of OIDCVCI.



3.3.3.7.5 Credential Issuer Provided Nonce

Upon receiving a Credential Request, the Credential Issuer *MUST* require the Wallet to send a proof of possession of the key material it wants a Credential to be bound to. This proof *MUST* incorporate a nonce generated by the Credential Issuer. The Credential Issuer will provide the client with a nonce in an error response to any Credential Request not including such a proof or including an invalid proof.

Below is a non-normative example of a Credential Response with the Credential Issuer requesting a Wallet to provide in a subsequent Credential Request a proof that is bound to a `c_nonce`:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Cache-Control: no-store

{
    "error": "invalid_or_missing_proof",
    "error_description":
        "Credential Issuer requires proof to be bound to a Credential Issuer
provided nonce.",
    "c_nonce": "8YE9hCnyV2",
    "c_nonce_expires_in": 86400
}
```

3.3.4 Authenticating with Verifiable Credentials

3.3.4.1 Overview

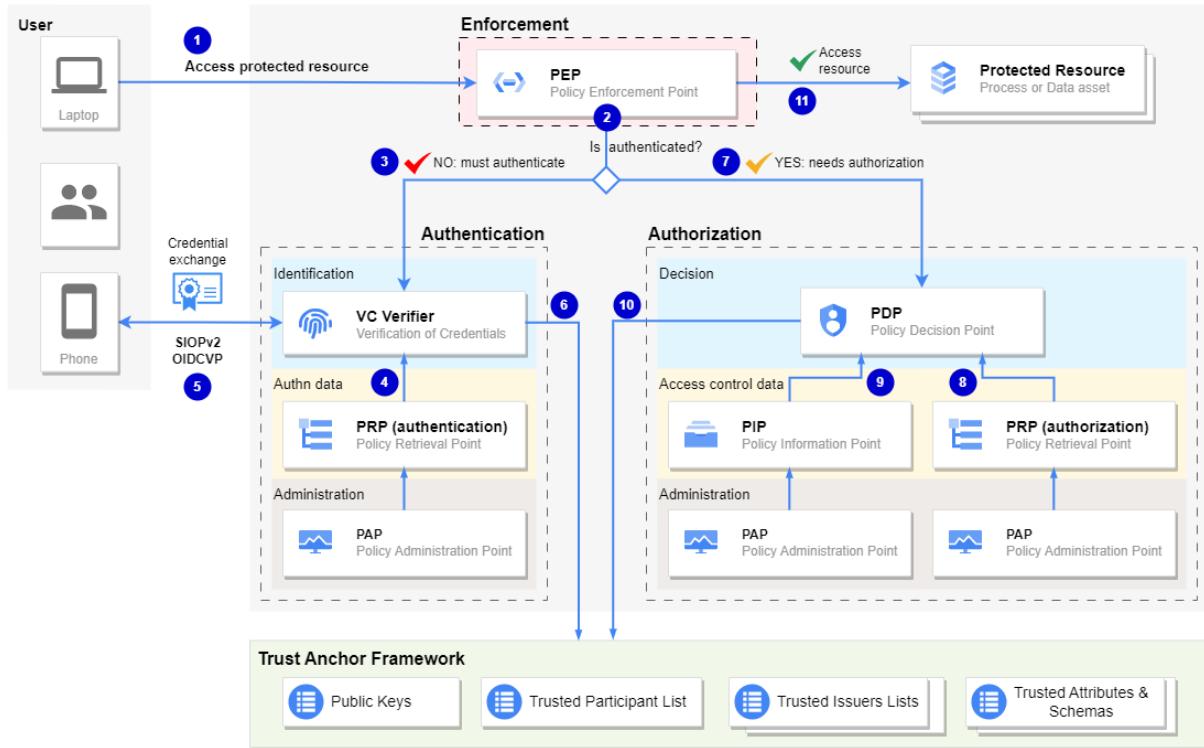
Authentication requires a special type of Verifiable Credential called a VerifiableID. For the examples illustrating the mechanism we will use a LEARCredential, which is at the same time a VerifiableId and a Verifiable Authorization.

For the explanation we will use the following figure which is based on [NIST Special Publication 800-178: A Comparison of Attribute Based Access Control \(ABAC\) Standards for Data Service Applications](#) which describes a reference architecture with necessary functional components to achieve authentication and enforcement of the authorization policies. The authorization process depends on four layers of functionality: **Enforcement, Decision, Access Control Data, and Administration**.

The diagram uses the logical concepts also used in the XACML reference architecture, but the description is agnostic to the actual policy language used in a concrete implementation.

In this description we assume that the user is a natural person who wants to access a protected resource. However, the architecture supports machine-to-machine (M2M) interactions. The differences will be described in the description when they are relevant.



Figure 22 - [Figure_Architecture_overview](#)

The high level flow is the following:

- 1 The user sends a request to access a protected resource managed by the Relying Party.
- 2 Every request to a protected resource is intercepted by the PEP (Policy Enforcement Point). The PEP is normally implemented as a reverse proxy or API gateway, so all calls to protected resources entering the organisation are intercepted and forwarded only if they are accepted. If the request does not come with authentication information, there are several high level scenarios which are possible:
 - o If the user is a machine using an API, the request is rejected with an error. The caller is responsible for authenticating before retrying, which can be done using the metadata about the Relying Party available at the standard OpenID endpoints.
 - o If the user is a natural person using a browser for navigating the portal of the Relying Party, the typical approach is to redirect the browser to the authentication component using an HTTP Status of 302. The redirection includes enough information so the authentication component can send back the user to the original request if the user completes authentication successfully.

Alternatively, the request can include authentication information which has been obtained before via an authentication process. If this is the case, the PEP goes directly into the authentication phase, described later in section [Access control with Verifiable Credentials](#).

3 The user is redirected to the authentication component. As mentioned before, alternatively the authentication component can be accessed directly by the user before trying to access the protected resource. This would be the expected behaviour of a user who knows that authentication is required. The mechanism described here supports all possible scenarios both for natural persons and machines. An example flow with a natural person would be: The user navigates first to a general Login page in the portal of the Relying Party and authenticates. After authentication the portal presents a collection of services that the user can access. The user selects one of the services/operations and the Relying Party executes the operation if the user is authorised to do so.

4 The Verifiable Credentials verifier component of the Relying Party retrieves the credentials that are required for authentication. If the user has been redirected when trying to access a protected resource, the redirection provides information about the operation and protected resource that was trying to access, so there may be more credentials required than just a VerifiableID.

If the user accessed the Login page directly, the VC Verifier component does not have information about the resource that the user ^{intends} to access, so it can request just a VerifiableID.

5 The VC Verifier component of the Relying Party and the Wallet of the user engage in a Verifiable Presentation Exchange flow, where the result is that the VC Verifier receives a VerifiableID and possibly additional Verifiable Credentials as determined in step 4.

6 The authorization mechanism uses the Trust Framework to query different Trusted Lists that are needed to evaluate the policy rules applicable to the request and compute a decision.

The Trust Framework is composed of one or more Trusted Lists where each list is specialised in some specific domain of trust. For example, the Trust Framework contains Trusted Lists with the schema definitions used for the well-known types of Verifiable Credentials in the ecosystem. In this way, if the Verifiable Registry used for the Trust Framework is based on a Blockchain network (or several federated networks), the schemas can be published in a trusted and resilient manner reducing the risk of malicious tampering and keeping a record of the history of modifications. But the most interesting Trusted Lists in the Trust Framework are the Trusted Issuers Lists, which provide the mechanism to efficiently and securely verify that the issuer of a given credential is a Trusted Issuer of that type of credentials. It has to be noticed that even though most Trusted Issuer Lists are global (used by all participants in the ecosystem) and have essentially public information, there may be some specialised lists that are private to one or more organisations and specialised in some specific requirement.

In this way, there may be some lists that are specific to the products that one organisation provides to other participants in the ecosystem. For example, there may be a Trusted Issuer List private to one organisation which is updated with the identity of a participant when that participant acquires access to some product provided by that organisation.

For example, this "product-specific" Trusted Issuer List is queried when performing authorization if the related policy rules specify that the issuer of the Verifiable Credential received in the request should have acquired the product that the request is trying to access.



However, as mentioned before, the set of "internal" and "external" Trusted Lists to query is not hardcoded but it depends on the specific policy rules that should be applied to the request when computing the decision to grant access or not. In this way, the system provides a great deal of flexibility on how access control is performed.

For simplicity, the diagram below shows the query of the Trusted Issuer Lists as a single interaction to a single box, but the actual interactions can be very complex if required. In addition, the diagram does not specify whether the list is "private" or "global".

The following sections elaborate in more detail in each of the interactions.

3.3.4.2 Starting the OpenID for Verifiable Presentations flow

In this section we assume that the user tries to access a protected resource and that the user is not authenticated. The flow when the user is already authenticated is the same, just skipping the authentication phase (steps 1 to 6).

The authentication process starts an OpenID for Verifiable Presentations flow combined with the Self-Issued OP v2 specification [[OpenID.SIOP2](#)], where the VC Verifier component of the portal plays the role of a Relying Party (RP in Open ID Connect terminology) and the wallet of the user is a Self-Issued IdP.

In this step, the Verifier has to send an Authorization Request to the wallet. But as a Self-Issued OP may be running locally as a native application or progressive web application (PWA), the RP may not have a network-addressable endpoint to communicate directly with the OP. We have to leverage the implicit flow of OpenID Connect to communicate with such locally-running Ops, as described in [[OpenID.SIOP2](#)].

We use a QR code to start the process and allow the wallet to receive the Authorization Request from the Verifier, and respond with an Authorization Response, sent to the Verifier in the body of an HTTP **POST** request.



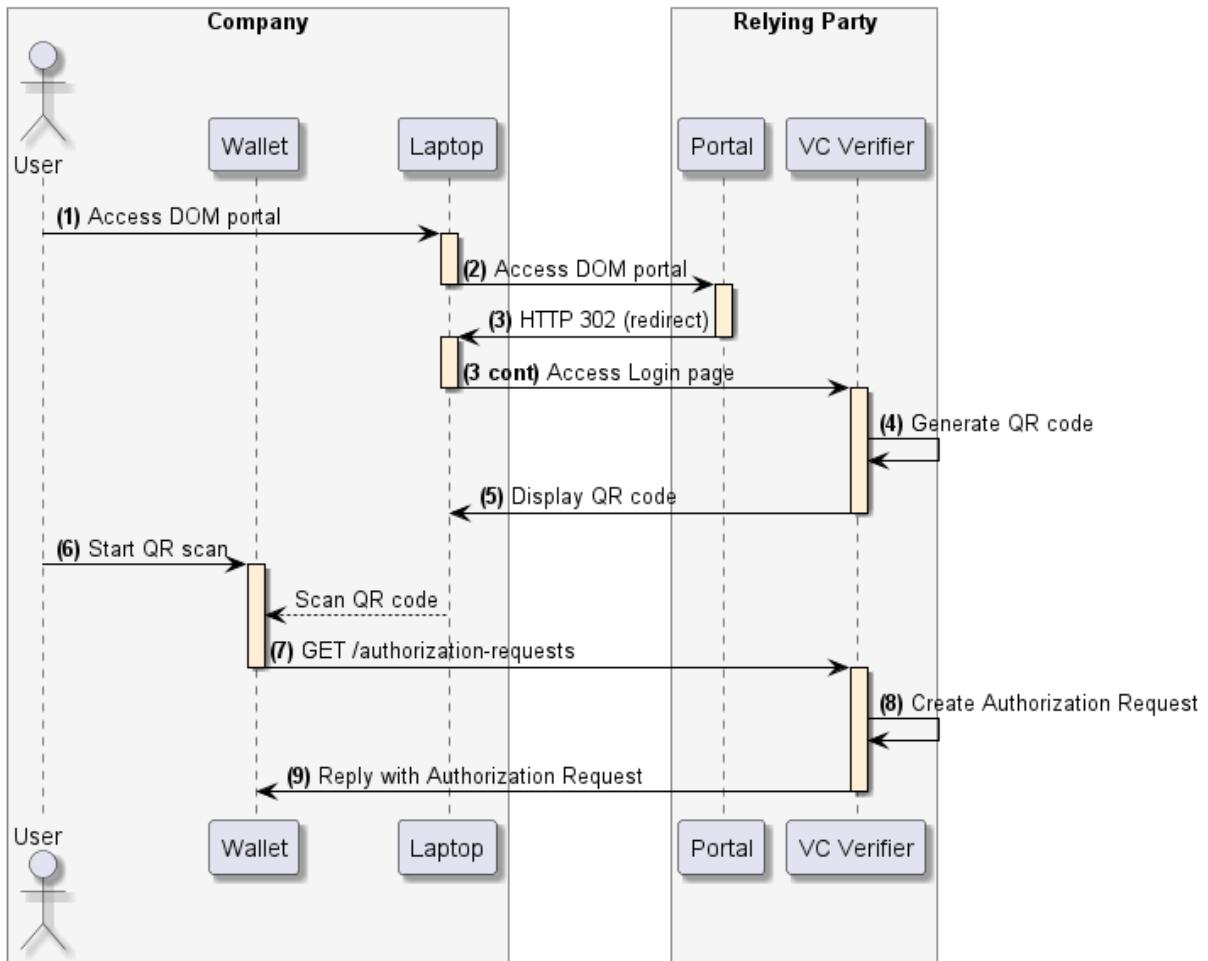


Figure 23 - Starting the authentication phase

1 The user (GoodAir employee) uses his Laptop browser to access the DOME portal to perform the onboarding process.

We assume in this scenario a cross-device interaction, that is, the user accesses the portal services with a PC browser, but authentication is performed with a mobile using Verifiable Credentials, as in typical 2-factor authentication.

2 The browser sends a request to the Packet Delivery portal server.

3 The portal redirects the user to the login page of the Verifier component of DOME Onboarding. The page shows a button labelled “Login with Verifiable Credentials” or something similar.

4 The Verifier generates a QR code, containing inside the URL of the `/authentication-requests` endpoint of the Verifier component which will be used to start the [OpenID.VP] process.

A QR code is used because a Self-Issued OP may be running locally as a native application or progressive web application (PWA), the RP may not have a network-addressable endpoint to communicate directly with the OP. We have to leverage the implicit flow of OpenID Connect to communicate with such locally-running Ops, as described in [OpenID.SIOP2].

```
https://verifier.dome-marketplace.eu/authorization-  
requests?state=af0ifjsldkj
```

- 5** The QR code is displayed in the user browser with instructions to scan it and go to the URL inside it.



Figure 24 - *Figure QR code with the example Authorization Request endpoint inside*

- 6** The user scans the QR with his wallet and tells the wallet to go to the URL in the QR.
7 After confirmation by the user, the wallet performs a **GET /authentication-requests** to the endpoint that was inside the QR code. The reply to the GET request is an Authorization Request
8 The VC Verifier creates a SIOP Authentication Request. The parameters comprising a request for verifiable presentations are described in detail in the next section
9 The Verifier replies to the wallet with the Authorization Request

3.3.4.3 Generating the Authorization Request

The Authorization Request travels in the response body of the HTTP GET request performed in the previous point, as a JWT signed by the DOME onboarding service, using the eIDAS certificate for seals that corresponds to the legal person operating the service and with the JAdES signature format.

The parameters comprising a request for verifiable presentations are given in section 5 of [OpenID.VP] and in section 10.1 of [OpenID.SIOP2] and are reproduced here with the particularities of this use case, in particular taking into account that this is a cross-device interaction:

- **response_type** (*REQUIRED*). Must be `vp_token`. This parameter is defined in [RFC6749]. The possible values are determined by the response type registry established by [RFC6749]. The [OpenID.VP] specification introduces the response type `vp_token`. This response type asks the Wallet to return only a VP Token in the Authorization Response, which is what we want in our case.
- **scope** (*REQUIRED*). This parameter is defined in [RFC6749] which allows it to be used by verifiers to request presentation of credentials by utilizing a pre-defined scope value designating the type of credential. See section [Request Scope](#) for more details. We use in this instance the value `dome.credentials.presentation.LEARCredential` which means that the RP (DOME onboarding service) is asking the SIOP (user wallet) to send a credential of type `LEARCredential`.
- **response_mode** *REQUIRED*. *MUST* be `direct_post`. As this is a cross-device scenario, this response mode is used to instruct the Self-Issued OP to deliver the result of the authentication process to a certain endpoint using the HTTP POST method. This endpoint to which the SIOP shall deliver the authentication result is conveyed in the parameter `redirect_uri` described below.
- **redirect_uri** (*REQUIRED*). *MUST* be a valid RP endpoint. The Authentication Response is sent to this endpoint using `POST` and encoding `application/json`.
- **client_id** *REQUIRED*. *MUST* be the DID of the RP (DOM onboarding entity) so it can be resolved by the SIOP and checked against a Trusted List, or rejected if it does not pass validation. This provides a high level of assurance to the SIOP that the RP is really who it claims.
- **client_id_scheme** *REQUIRED*. *MUST* have the value `did`. This value indicates that the Client Identifier is a DID defined in [DID-Core]. The request *MUST* be signed with a private key associated with the DID. To obtain the corresponding private key, the Wallet *MUST* use DID Resolution defined by the DID method used by the Verifier. For most DID methods and since the associated DID Document may include multiple public keys, a particular public key used to sign the request in question *MUST* be identified by the `kid` in the JOSE Header. However, in our case the Verifier uses `did:elsi` and so the request *MUST* be signed with the eIDAS certificate according to the [ETSI-JADES] format, which defines the mechanism to identify the public key. All Verifier metadata other than the public key *MUST* be obtained from the `client_metadata` or the `client_metadata_uri` parameter as defined in Section 5 of [OpenID.VP].
- **nonce** *REQUIRED*. This parameter follows the definition given in [OpenID.Core]. It is used to securely bind the verifiable presentation(s) provided by the wallet (SIOP) to the particular transaction managed by the RP.
- **state** *REQUIRED*. Used by the portal component of DOM onboarding to associate the start of an authentication session with the end of that session when the RP Verifier component notifies to the portal.
- **presentation_definition** *CONDITIONAL*. A string containing a `presentation_definition` JSON object as defined in Section 4 of [DIF.PresentationExchange]. We do not use this parameter because `scope` already specifies the credential type.
- **presentation_definition_uri** *CONDITIONAL*. A string containing a URL pointing to a resource where a `presentation_definition` JSON object as defined in Section 4 of [DIF.PresentationExchange] can be retrieved. We do not use this parameter because `scope` already specifies the credential type.



Note: A request *MUST* contain either a `presentation_definition` or a `presentation_definition_uri` or a single `scope` value representing a presentation definition, those three ways to request credential presentation are mutually exclusive. We use here the `scope` mechanism, which is simpler and fits our use case.

This is an example request object using the definitions above:

```
openid://?
scope=dome.credentials.presentation.LEARCredential&
response_type=vp_token&
response_mode=direct_post&
client_id=did:elsi:VATFR-99999999&
redirect_uri=https://verifier.dome-marketplace.eu/api/authentication_response&
state=af0ifjsldkj&
nonce=n-0S6_WzA2Mj
```

(URL encoding removed, line breaks and leading spaces added for readability).

As mentioned above, the Authentication Request is returned to the wallet in the reply body of the GET request, as a JWT in JWS form [RFC7515]), signed with eIDAS certificate of the DOM onboarding legal person.

This is an example of the unencoded contents of the payload of the JWT:

```
{
  "iss": "did:elsi:VATFR-99999999", // Should correspond with the client_id
  in the AR
  "sub": "did:elsi:VATFR-99999999", // Should correspond with the client_id
  in the AR
  "aud": "https://self-issued.me/v2", // As specified in section 5.5 of
  OIDC4VP
  "iat": 1667194901,
  "exp": 1667194961, // To avoid replays, here it expires in
  60 secs
  "auth_request": "openid://?scope=...&response_type=vp_token&..."}
```

Where the claims `iss` and `sub` *MUST* be the DID of the RP (in this case DOME onboarding) and *MUST* correspond exactly with the `client_id` parameter in the Authorization Request. The claim `auth_request` contains the Authentication Request as a string. The expiration time in claim `exp` avoids replays and can be very short because it is used by the Wallet just on reception of the Authentication Request. The expiration time should be enough for the user to review the Authorization Request, decide the credential(s) to send to the RP and instruct the Wallet to send the Authorization Reply to the RP.



3.3.4.4 Verification of the Authorization Request

Before sending the Authentication Response, the wallet should verify that the Authorization Request is correct. The most important verifications are described here:

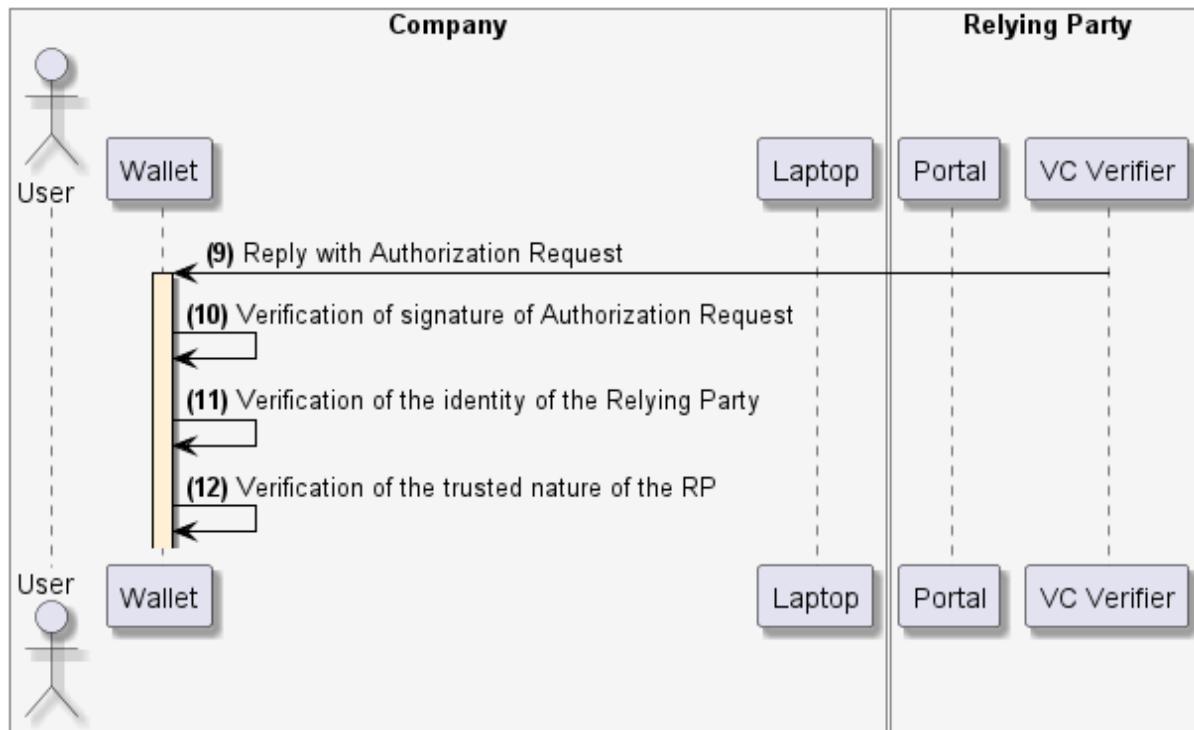


Figure 25 - [Figure Starting the authentication phase](#)

10 Verification of the signature. As mentioned in the previous section, the Authorization Request is received as a JWT signed by the DOME onboarding service, using the eIDAS certificate for seals that corresponds to the legal person operating the service. This allows the wallet to verify that the signature corresponds to a real-world entity.

11 Verification that the DID in the `client_id` field of the Authorization Request corresponds to the entity that signed the Authorization Request. This is easy in our case because we use the `did:elsi` method.

12 Verification that the entity identified in the `client_id` field of the Authorization Request is a trusted entity belonging to the ecosystem, by resolving the DID in the `client_id` field. The actual mechanism may vary depending on the DID method used and the ecosystem where the client wants to onboard.

If the Relying Party is a well known entity (like the DOME onboarding service in the case of DOME), it is easy to verify that the DID corresponds to the Relying Party. In most other cases, for example when the Relying Party is a participant in the ecosystem, we assume that the ecosystem provides a Trusted Participants Registry with an API compatible with the EBSI Trusted Issuers Registry, and that the participants registry is managed in a trusted way by the onboarding service (meaning that the Wallet user trusts on the onboarding service, in the same way as all other participants).

In this case, to check if the DID is a participant on the ecosystem, the wallet sends a **GET /api/did/v1/identifiers/{did-to-verify}** request to the endpoint of one of several trusted servers implementing the functionality to query the Trusted Participant Registry, where **{did-to-verify}** is the actual DID the Wallet wants to check (in our example it would be **did:elsi:VATFR-99999999**, corresponding to the RP). The API returns a JSON document as a DID Document. The DID Document (as per W3C) contains relevant information about the entity owner of the DID. It contains its Public Key, used to verify the digital signature of the entity. It also contains the status of the entity in the ecosystem. It is extensible and can contain any public information which may be relevant for the use case. The API must be operated by a trusted entity for the Wallet user. There may be as many servers implementing the API as needed and operated by different entities. At least one of those trusted entities has to be configured in the Wallet of the user, to facilitate the use of the wallet.

3.3.4.5 Creating and sending the Authorization Response

Once the Authorization Request has been validated, the Wallet creates an Authorization Response to be posted in the **redirect_uri** specified by the RP in the Authorization Request. For completeness, we describe in the following figure the complete process from reception of Authorization Request until sending the Authorization Response.

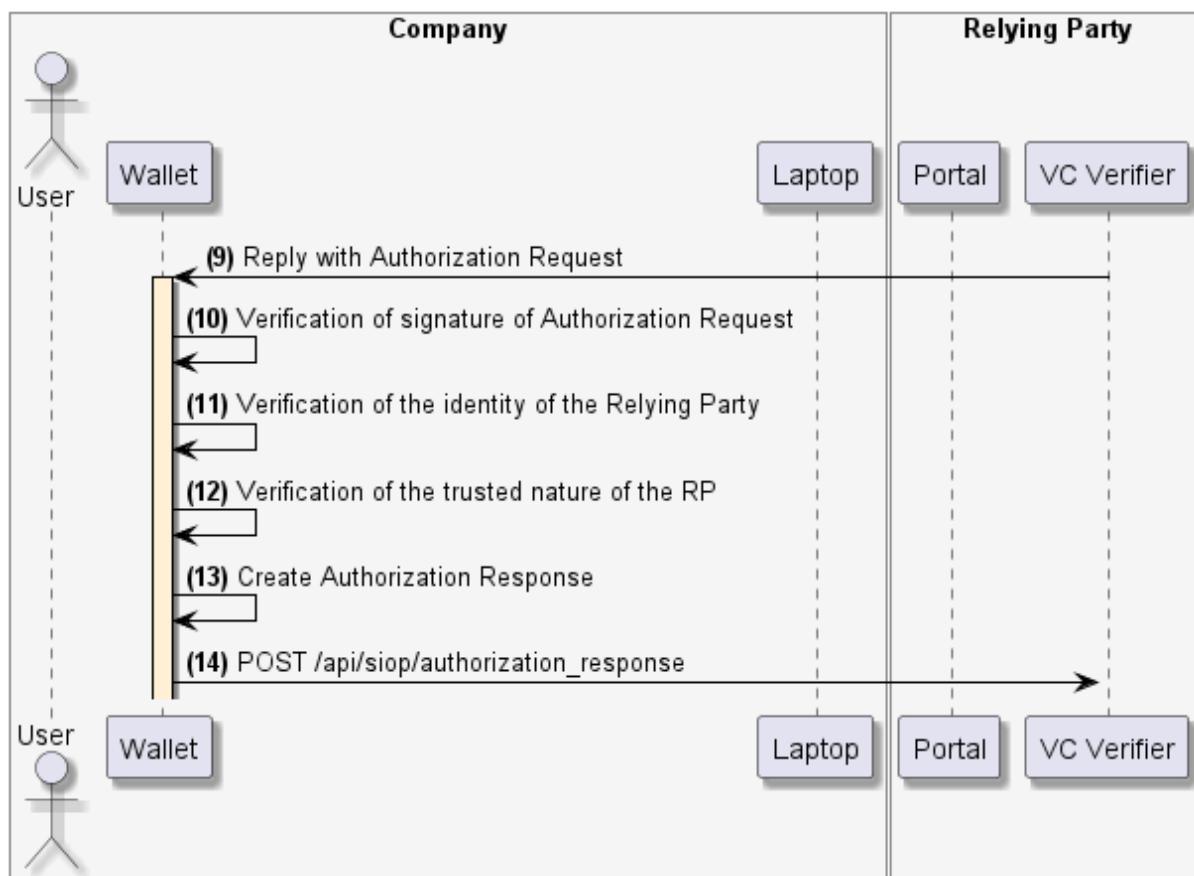


Figure 26 - *Figure Starting the authentication phase*



13 The wallet creates an Authorization Response to be posted in the `redirect_uri` specified by GoodAir in the Authorization Request that was sent to the wallet. The contents of the Authorization Response are described below. The response is constructed as defined in section 6.1 of [OpenID.VP]. In particular, because the Authorization Request included only `vp_token` as the `response_type`, the VP Token is provided directly in the Authorization Response and a separate `id_token` is not needed. The contents of the Authorization Response in our specific use case are:

```
presentation_submission=[see definition below]

&vp_token=[see definition below]
```

The content of the `presentation_submission` parameter in the above Authorization Response is:

```
{
  "definition_id": "OnboardingPresentationDefinition",
  "id": "OnboardingPresentationSubmission",
  "descriptor_map": [
    {
      "id": "id_credential",
      "path": "$",
      "format": "ldp_vp",
      "path_nested": {
        "format": "ldp_vc",
        "path": "$.verifiableCredential[0]"
      }
    }
  ]
}
```

Which complies with [DIF.PresentationExchange] and refers to the Verifiable Presentation in the `vp_token` parameter provided in the same response. In our example, the Verifiable Presentation includes the LEARCredential that was described in the previous sections.

14 The wallet sends the Authorization Response to the endpoint received in the `redirect_uri` parameter of the Authorization Request, sending an HTTP `POST` request using the encoding `application/x-www-form-urlencoded`.



```
POST /api/siop/authorization_response HTTP/1.1
Host: verifier.dome-onboarding.org
Content-Type: application/x-www-form-urlencoded

presentation_submission=[see definition below]

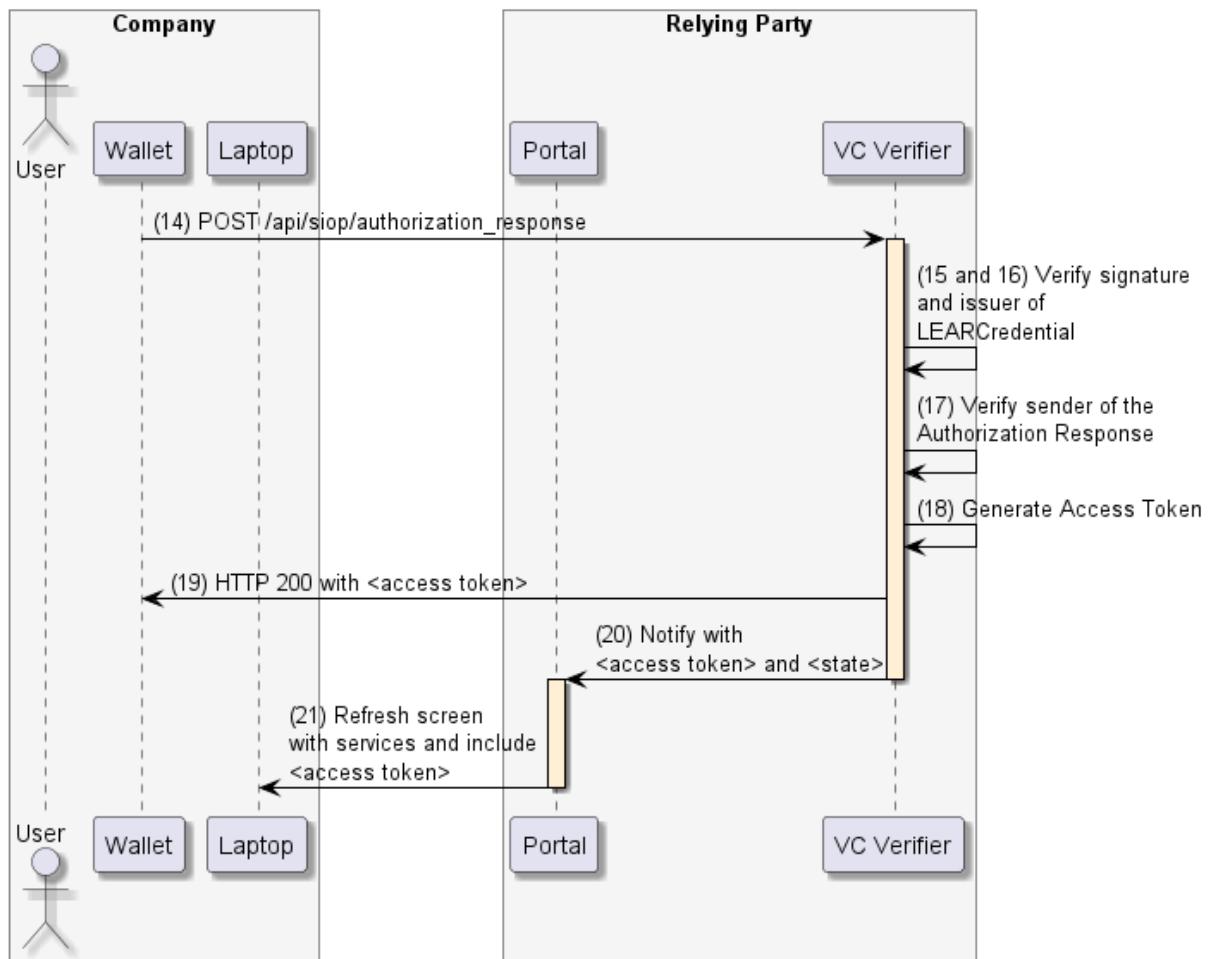
&vp_token=[see definition below]
```

3.3.4.6 Authenticating the user with the LEARCredential

We should remember that inside the LEARCredential the `credentialSubject` object has an `id` field with value `did:key:z6MkhaXgBZDvotDkL5257faiztiGiC2QtKLGpbnnEGta2doK` which is the DID of the user. It was generated during the creation of the LEARCredential, and the Relying Party should trust that the credential was generated in the proper way binding cryptographically the DID with the LEARCredential, as described in a previous section. It is the same trust that should be put in the generation of any other signed document, like a contract or invoice.

The process of verifying the Authorization Response and authenticating the user with the LEARCredential is the following:



Figure 27 - [Figure Starting the authentication phase](#)

15 and 16 The Verifier receives the Authorization Request and has to perform verifications, the standard ones being defined in [DIF.PresentationExchange]. In order to verify that the Verifiable Credential has been issued by a trusted entity, the Relying Party has to verify:

- That the DID of the entity which is the issuer of the VC is a trusted entity.
- That the VC was signed by that participant.

Both verifications can be done by performing DID resolution, checking that the resulting DID Document contains the public key corresponding to the one specified in the Verifiable Credential, and by verifying the digital signature of the credential against that public key. In our case the LEARCredential was issued using the `did:elsi` method, so resolution is very simple and is specified in [DID-ELSI].

17 After verifying the credential, the Relying Party can also verify that the Verifiable Presentation including the Verifiable Credential is sent by the user and not by a malicious agent. To do so, it uses the public key associated to the `did:key` identifier `id` field inside the `credentialSubject` structure. That public key is cryptographically bound to the customer DID during the onboarding process that GoodAir performed with its employee.

18 The Verifier creates an Access Token for the user so it can be used later for access services provided by the Relying Party (in the case of onboarding, those services would be the ones related to the onboarding process implemented by the Onboarding

portal). The Access Token is generated in JWT format and signed by the VC Verifier. The access token is intended for use as bearer token over HTTP [RFC2616] using Transport Layer Security (TLS) [RFC5246] to access protected resources, and it should use the JWT Profile profile described in [RFC9068]. For our use case, the payload of the JWT access token looks like:

```
{
  "iss": "did:elsi:VATFR-99999999",
  "sub": "did:key:z6MkhaXgBZDvotDkL5257faiztiGiC2QtKLGpbnnEGta2doK",
  "aud": "https://dome-marketplace.eu/onboarding",
  "exp": 1639528912,
  "iat": 1618354090,
  "jti": "dbe39bf3a3ba4238a513f51d6e1691c4",
  "client_id": "did:elsi:VATFR-99999999",
  "scope": "vp_token",
  "verifiableCredential": ["the VC that was received inside the Verifiable Presentation"]
}
```

Where, according to [RFC9068]:

- **iss** and **client_id** have the same value because the access token has been generated by the Verifier component of the DOME onboarding service, acting as the DOME legal person.
- **sub** identifies the user (employee of GoodAir acting as LEAR of the company) using the DID inside the Verifiable Credential received
- **aud** identifies the RS (Resource Server) component in Packet Delivery. In this case we assume that it is internal and does not have a DID assigned, so we use the URI of the component.
- **kid** in the header identifies the key that is used to sign the JWT and which must be configured up-front and known by the RS (Resource Server).
- **scope** has the same value as the equivalent **scope** parameter in the initial Authorization Request.
- **verifiableCredential** contains the Verifiable Credential that was received, specifically the value of the first element of the field **verifiableCredential** of the Verifiable Presentation.

19 The Verifier sends a successful response to the POST request from the wallet. The wallet receives the response to the POST indicating the success or failure of the process. The Verifier continues processing because the web portal of Packet Delivery has to be refreshed with the services that this specific user can access, based on the information received in the Verifiable Credential and the access control policies implemented by the DOME portal.

20 The Verifier notifies the DOME portal to refresh the login page so it can present the services to the user. The notification includes the following:

- The **state** nonce that was generated by the portal when it generated the QR code. The portal uses the **state** nonce to know what login session is being notified.



- The access token generated before.

The notification is a simple POST request with both parameters in the body:

```
POST /api/notify HTTP/1.1
Host: dome-marketplace.eu
Content-Type: application/x-www-form-urlencoded

access_token=[the access token]

&state=af0ifjsldkj
```

The portal component replies immediately and continues processing.

21 The portal of the Relying Party refreshes the screen and displays the services available to users, sending the Access Token to the browser of the user. The Access Token will be sent back by the browser to the portal whenever the user tries to access a protected resource of the portal, and so access control can be performed using the data inside the token, in the same way as with any other access token in other authentication mechanisms.

3.3.5 Access control with Verifiable Credentials

3.3.5.1 Overview

This section assumes that authentication has already been performed and that the Relying Party receives a request to a protected resource, with a proper access token inside the request. If the request does not come with an access token, the request is rejected immediately (or the user is redirected to the Authentication service of the Relying Party, depending on the use case).

The Access Control mechanism described here is the same independently of whether the request comes from a user or a machine, so the M2M use case is supported without any modification. In the following description we will use the term **user** interchangeably for both the user as a natural person or the user as a machine.

However, the assumption is that all machines have an identity (they are assigned an identifier), and that they are acting **on behalf of** a legally valid identity in the sense of eIDAS2. In other words, we require that the identities of users and machines can be cryptographically bound to the real-world identity of either a natural person or a legal person in a way that provides legal certainty to the Relying Party regarding the chain of responsibility.

If the user is a natural person using the portal of the Relying Party, the request to the protected resource will come from the browser that the user employs to navigate the portal, and the access token will be included automatically by the browser in every request that is sent to the Relying Party, for example when the user clicks a button or fills a form.



If the user is a server or device, it is assumed that it has obtained an access token previously as the result of an authentication process described above. The server or device is responsible for including the access token in every request that is sent to the Relying Party to access a protected resource.

If the portal of the Relying Party is an onboarding process (e.g., the portal of the DOME Onboarding service), the protected resource could be the onboarding service itself, or a service to upload additional information for a partially completed onboarding process.

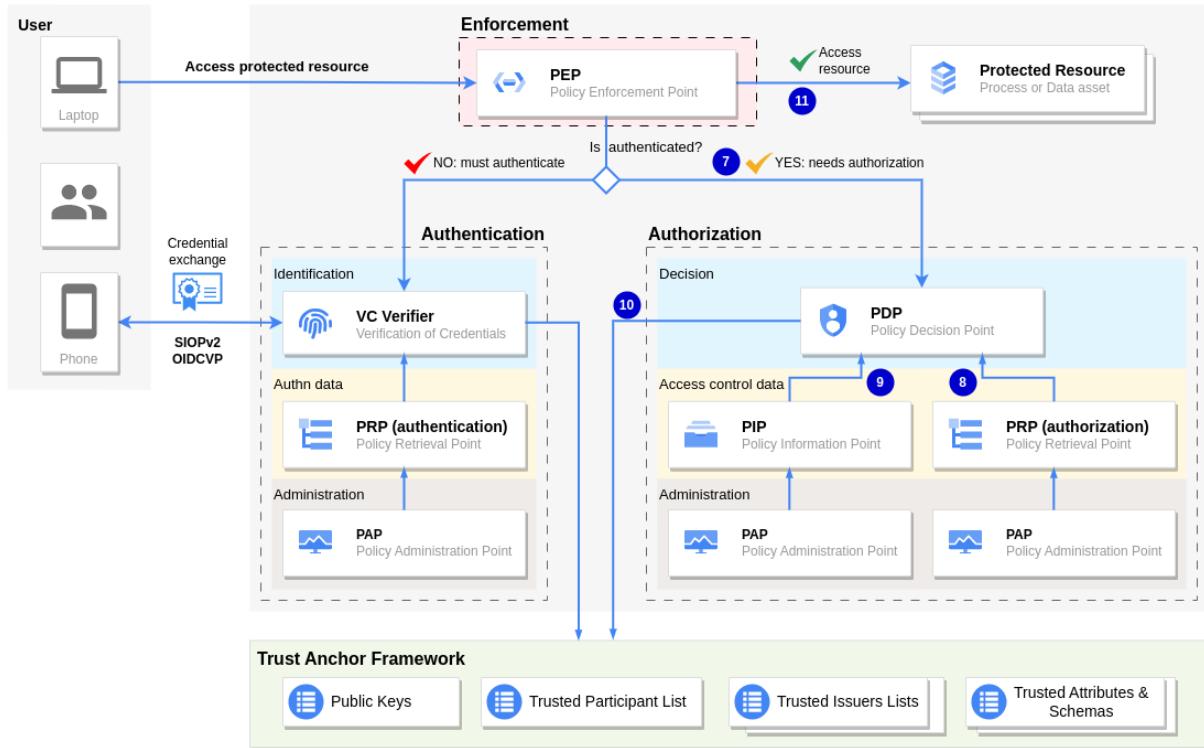
If the portal is a Marketplace and the organisation using the portal wants to offer products in it, these services could be to register the organisation as a *Seller*, or to create a Product specification and offering.

If the portal is the Marketplace and the organisation wants to contract products in it, these services could be to register the organisation as a *Customer*, or to launch a procurement order of a product.

The same mechanism can be used by any participant in the ecosystem, if they want to use an advanced IAM mechanism which is aligned with the EU strategy on digital identity based on Verifiable Credential and eIDAS2.

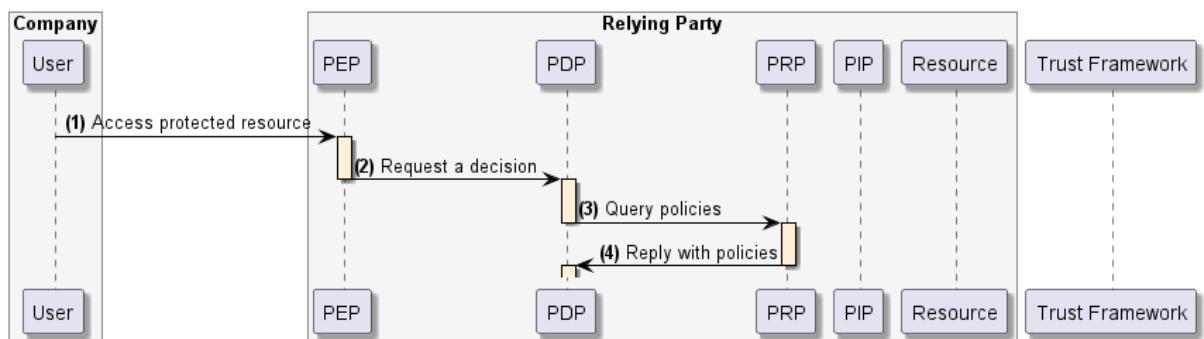
For the explanation we will use the following figure which is based on [NIST Special Publication 800-178: A Comparison of Attribute Based Access Control \(ABAC\) Standards for Data Service Applications](#) which describes a reference architecture with necessary functional components to achieve enforcement of the authorization policies. The authorization process depends on four layers of functionality: **Enforcement**, **Decision**, **Access Control Data**, and **Administration**.



Figure 28 - [Figure High level Authentication logical architecture](#)

3.3.5.2 Determine the Authorization Policies which apply to the request

Every request to a protected resource is intercepted by the PEP (Policy Enforcement Point), and to know if the request should be authorised or not, the PEP asks to the PDP (Policy Decision Point) for a decision. The PDP needs to know what are the policy rules that should be applied to the request, and it uses the PRP (Policy Retrieval Point) to retrieve those policies.

Figure 29 - [Figure Determine the Authorization Policies](#)

- 1** The user (natural person or machine) sends a request to access a protected resource, and the PEP (Policy Enforcement Point) component intercepts the call. The PEP is normally implemented as a reverse proxy or API gateway, so all calls to protected resources entering the organisation are intercepted and forwarded if they are accepted.



2 If the request is not authenticated it is rejected with an error. Depending on the use case, the error returned can contain additional information that enables the user agent to be redirected to perform authentication.

After inspecting the request, the PEP requests a decision from the PDP passing a series of attributes. In general, the request from the PEP to the PDP consists of **subject** attributes (typically for the user who issued the request), **resource** attributes (the resource for which access is sought), **action** attributes (the operations to be performed on the resource), and **environment** attributes.

Some of the above attributes are obtained by the PEP from the original request received from the user. For example, the HTTP verb (GET, POST, PUT, PATCH, DELETE) and the URL and possibly the body of the request provide action attributes. Specifically, in the case of an NGSI-LD request those attributes are formally specified in the NGSI-LD standard. The subject attributes most relevant for access control are located in the access token received with the request. In particular, the access token includes the Verifiable Credentials that were employed for authentication, and the claims in those credentials can be used to evaluate access control policies. This includes (but is not limited to) the roles object that may be included in the VerifiableID credential used for authentication.

An example access token would be:

```
{  
  "token_type": "Bearer",  
  "expires_in": 3600,  
  "access_token": "ewogICJhbGciOiAiR...ERBIgogIH0KfQ"  
}
```

Decoding the `access_token` field we could see a Verifiable Presentation including the LEARCredential that the user presented when authenticating:



```
{
  "@context": [
    "https://www.w3.org/ns/credentials/v2",
    "https://dome-marketplace.eu/2022/credentials/learcredential/v1"
  ],
  "id": "urn:did:elsi:25159389-8dd17b796ac0",
  "type": ["VerifiableCredential", "LEARCredential"],
  "issuer": {
    "id": "did:elsi:VATES-12345678"
  },
  "issuanceDate": "2022-03-22T14:00:00Z",
  "validFrom": "2022-03-22T14:00:00Z",
  "expirationDate": "2023-03-22T14:00:00Z",
  "credentialSubject": {
    "id": "did:key:z6MkhaXgBZDvotDkL5257faiztiGiC2QtKLGpbnnEGta2doK",
    "title": "Mr.",
    "first_name": "John",
    "last_name": "Doe",
    "gender": "M",
    "postal_address": "",
    "email": "johndoe@goodair.com",
    "telephone": "",
    "fax": "",
    "mobile_phone": "+34787426623",
    "legalRepresentative": {
      "cn": "56565656V Jesus Ruiz",
      "serialNumber": "56565656V",
      "organizationIdentifier": "VATES-12345678",
      "o": "GoodAir",
      "c": "ES"
    },
    "rolesAndDuties": [
      {
        "type": "LEARCredential",
        "id": "https://dome-marketplace.eu//lear/v1/6484994n4r9e990494"
      }
    ]
  }
}
```

Environmental attributes, which depend on the availability of system sensors that can detect and report values, are somewhat different from subject and resource attributes, which are administratively created. Environmental attributes are not properties of the subject or resources, but are measurable characteristics that pertain to the operational or situational



context in which access requests occur. These environmental characteristics are subject and resource independent, and may include the current time, day of the week, or threat level.

The PDP computes decisions to permit or deny subject requests to perform actions on resources. In computing a decision, the PDP queries policies stored in a PRP (Policy Retrieval Point).

If the attributes of the request are not sufficient for rule and policy evaluation, the PDP may need to search the PIP for additional attributes. The PIP is shown as one logical store, but in fact may comprise multiple physical stores of different types. In addition to the PIP, the PDP uses one or more Trusted Issuers Lists (to check if the issuer of the credential is included in them as a trusted issuer of that type of credential), and also the Authorization Registry (AR), which can check for finer granularity authorisation records which may be very dynamic in nature (e.g. if some resource usage by employees and machines owned by the issuer have exceeded some quota this month).

The actual Trusted Issuers Lists that the PDP needs to check is determined by the policy rules retrieved from the PRP. Those policy rules also determine the additional queries performed to the PIP data stores.

The set of information and data stored in the PIP and PRP comprise the access control data and collectively define the current authorization state, which the PDP will use to compute its decision.

3 The PDP asks the PRP for the policy rules applicable to this request. The policy rules may be specified using different policy languages, like ODRL, XACML or Rego. It is very difficult to make the interface between the PDP and the PRP completely transparent and independent from the concrete policy language used in an implementation. However, The PDP can be modularised in order to minimise the dependency and reduce the cost of migration to other policy language in the future.

4 The PDP receives the policy rules to apply for the current request.

3.3.5.3 Determine which Trusted Issuer Lists should be queried

After retrieving the policies from the PRP, the PDP determines what are the attributes needed by the policy rules to perform its evaluation. Some of those attributes can be determined from the request received (e.g., claims inside the Verifiable Credential(s) in the access token). But in general, the policy rules require additional attributes that have to be retrieved from other sources.

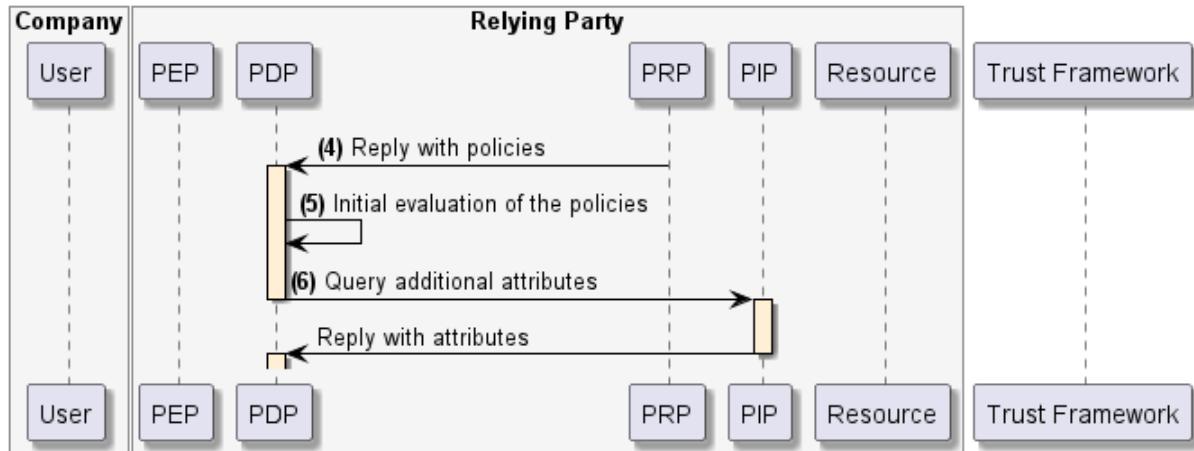
The two logical sources of those attributes are the PIP and the Trusted Issuers Lists in the Trust Anchor Framework.

The concept of the PIP is the same as in the standard XACML logical architecture, essentially enabling the PDP to retrieve environmental attributes required for policy evaluation.

In our approach using Verifiable Credentials for authentication and authorization, we use a Trust Framework to determine if the credentials received in the request are issued by authorised issuers of those credentials.

The PDP does not have hardcoded the actual Trusted Lists to query, but they are determined dynamically from the policies retrieved from the PRP. In simple use cases there may be one or two lists to query which are the same for all requests. But the mechanism described here can cater for very complex scenarios where the lists may be different depending on the requests and also from the dynamic environmental information retrieved from the PIP.



Figure 30 - [Figure Determine Trusted Issuer Lists to query](#)

- 4 The PDP receives the policy rules to apply for the current request.
- 5 The PDP performs an initial evaluation of the policy rules and if the attributes of the request are not sufficient for rule and policy evaluation, it determines the PIP stores to query, and also the set of Trusted Issuers Lists to query. The concrete Trusted Lists to query are dynamic and depend on the specific policies associated with the request.
- 6 Based on the policies and the attributes of the request, the PDP queries the PIP for additional attributes needed for the policy evaluation. Now that the PDP has the policy rules to apply and the attributes from the request and the ones retrieved from the PIP, the PDP can determine the actual Trusted Lists that have to be queried in order to be able to evaluate the policies.

3.3.5.4 Query the Trusted Issuer Lists

The authorization mechanism uses the Trust Framework to query different Trusted Lists that are needed to evaluate the policy rules applicable to the request and compute a decision.

The Trust Framework is composed of one or more Trusted Lists where each list is specialised in some specific domain of trust.

For example, the Trust Framework contains Trusted Lists with the schema definitions used for the well-known types of Verifiable Credentials in the ecosystem. In this way, if the Verifiable Registry used for the Trust Framework is based on a Blockchain network (or several federated networks), the schemas can be published in a trusted and resilient manner reducing the risk of malicious tampering and keeping a record of the history of modifications.

But the most interesting Trusted Lists in the Trust Framework are the Trusted Issuers Lists, which provide the mechanism to efficiently and securely verify that the issuer of a given credential is a Trusted Issuer of that type of credentials.

It has to be noticed that even though most Trusted Issuer Lists are global (used by all participants in the ecosystem) and have essentially public information, there may be some specialised lists that are private to one or more organisations and specialised in some specific requirement.



In this way, there may be some lists that are specific to the products that one organisation provides to other participants in the ecosystem. For example, there may be a Trusted Issuer List private to one organisation which is updated with the identity of a participant when that participant acquires access to some product provided by that organisation.

For example, this "product-specific" Trusted Issuer List is queried when performing authorization if the related policy rules specify that the issuer of the Verifiable Credential received in the request should have acquired the product that the request is trying to access.

However, as mentioned before, the set of "internal" and "external" Trusted Lists to query is not hard coded but it depends on the specific policy rules that should be applied to the request when computing the decision to grant access or not.

In this way, the system provides a great deal of flexibility on how access control is performed.

For simplicity, the diagram below shows the query of the Trusted Issuer Lists as a single interaction to a single box, but the actual interactions can be very complex if required. In addition, the diagram does not specify whether the list is "private" or "global".

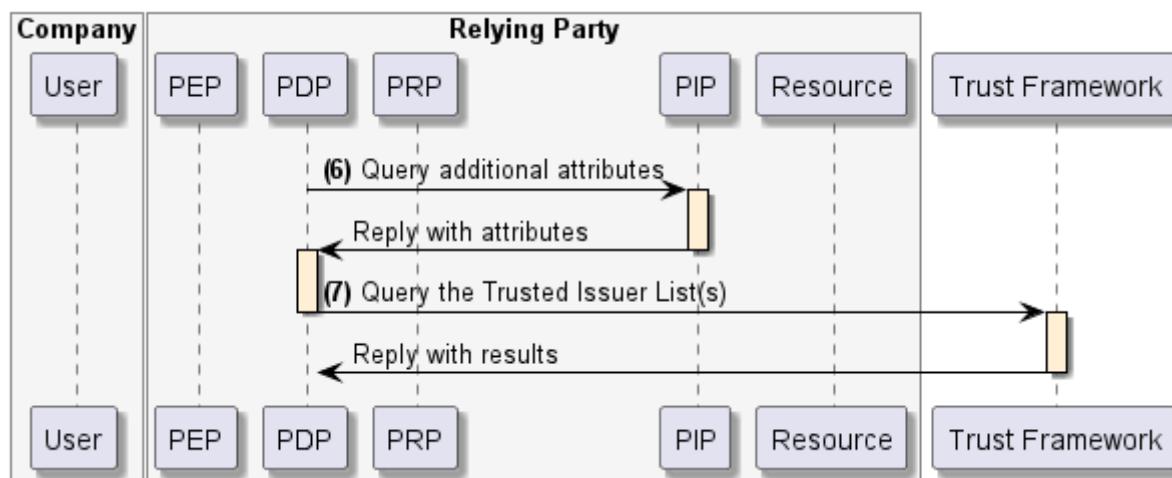


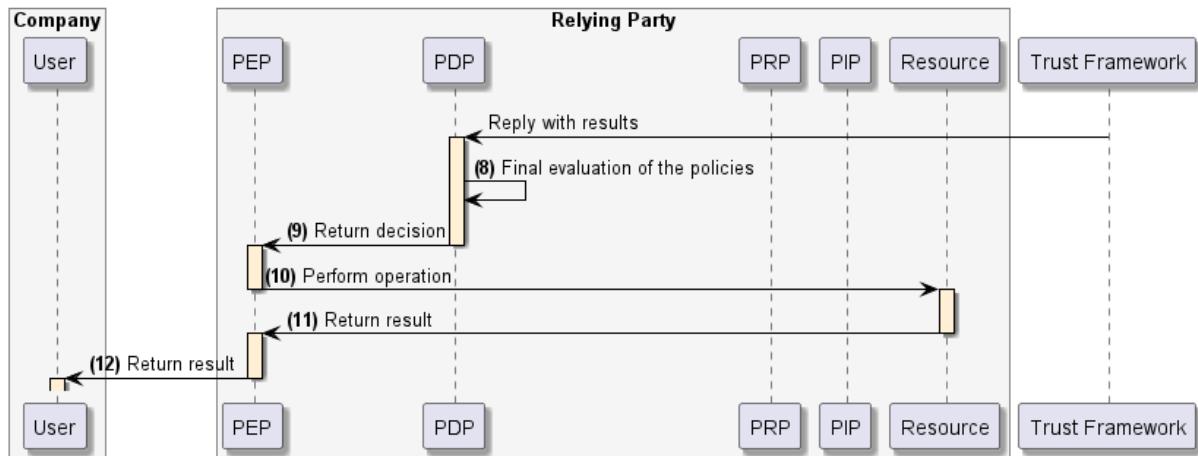
Figure 31 - [Figure Query the Trusted Issuer Lists](#)

6 Based on the policies and the attributes of the request, the PDP queries the PIP for additional attributes needed for the policy evaluation.

7 Based on the policies and the attributes of the request, the PDP queries one or more Trusted Issuer Lists to check that the credentials received comply with the requirements specified in the policies.

3.3.5.5 Compute the decision and allow (or not) access

Once the PDP has all the attributes required for policy evaluation, it computes a decision and returns that decision to the PEP which will enforce it.

Figure 32 - [Figure Compute the decision](#)

8 The PDP performs a final evaluation of the policy rules, taking into consideration the attributes received from the PEP, the attributes received from the PIP, and the Trusted Issuers Lists queried. After the evaluation the PDP computes a decision, either to accept or to reject the request.

9 The PDP returns the decision to the PEP, which will enforce it.

10 If the decision from the PDP is to reject the request, the PEP returns an error to the user. Otherwise, it forwards the original request to the Resource for execution.

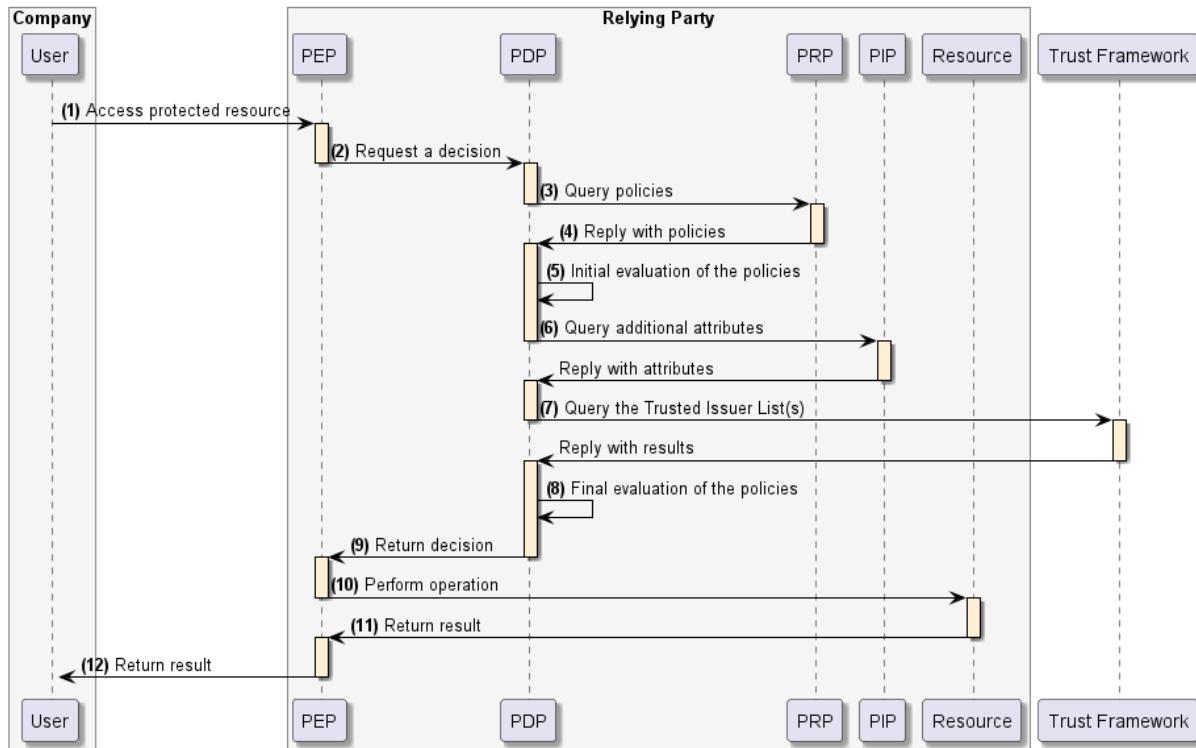
11 The resource executes the request and returns the result.

12 The PEP returns the result to the user.

3.3.5.6 Summary of the authorization process

The following diagram combines all the steps in a single one to provide an overall view of the authorization process.



Figure 33 - [Figure Summary of the Authorization process](#)

3.4 Detailed workflows illustrating interactions among components based on a reference use case

3.4.1 Access to DOME functions by DOME users

3.4.1.1 Explaining the Reference Use Case

Here we introduce the actors of the reference use case we are going to have. The main actors are marked in yellow in the following diagram.

RealTruth is a Trust Service Provider operating under the eIDAS Trust Framework, which issues a certificate for seals to the company GoodAir. GoodAir is a business that provides some services using an Air Quality application operated by GoodAir.

RealTruth also provides a certificate for signatures to the COO of the GoodAir company, so the COO can use the certificate to sign documents on behalf of GoodAir (like contracts, invoices, financial reports, ...) in a way that is compliant with eIDAS.



The COO of GoodAir will issue a verifiable credential of type LEARCredential to an employee of GoodAir, signing the credential with his certificate for signatures.

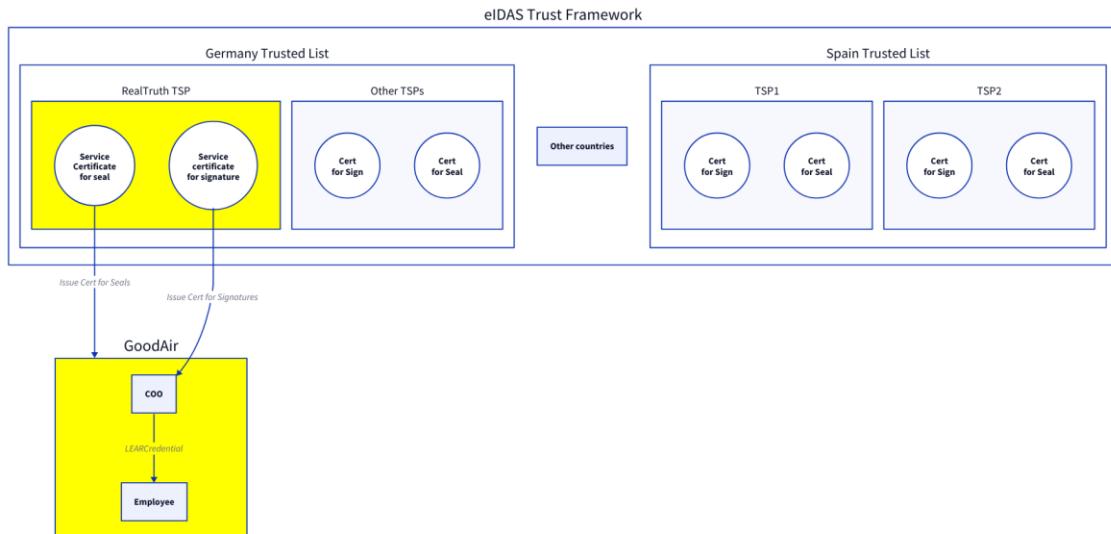


Figure 34 - SmartCityDS: an instance of a Data Space for Smart Cities

3.4.1.1.2 GoodAir: the company that will perform the onboarding process

The new participant is a business providing an Air Quality Monitoring application as a service. The business is called GoodAir and it operates the application, which receives data from a set of sensors that may or may not be the property of GoodAir. The sensors must have received a certification to be able to operate and send data to the GoodAir application.

The company is registered in the tax agency and business registry of Spain with VAT number VATES-12345678

3.4.1.1.3 Jesus Ruiz: COO (Chief Operating Officer) of GoodAir

The GoodAir company is small and the only person that can sign contracts on behalf of the company is the COO (Chief Operating Officer). The COO is a legal representative of the company and she is registered as so in the business registry of Spain.

The COO has a certificate for electronic signatures issued by one of the TSPs in Spain enabling the COO to perform electronic qualified signatures as legal representative of GoodAir, instead of doing them manually.

The X.509 certificate of the COO has the following contents in the **Subject** field (the example below is derived from a real certificate but with the identifiers modified to be an example):

```
cn=56565656V Jesus Ruiz
serialNumber=56565656V
givenName=Jesus
sn=Ruiz
```



2.5.4.97=VATES-12345678

o=GoodAir

c=ES

2.5.4.13=Notary:Juan Lopez/Protocol Num:7172/Date:07-06-2021

The above set of fields bind the legal identity of the COO with the identity of the business:

- **serialNumber** is the unique number recognized by the Spanish Government for spanish citizens (called NIF).
- **2.5.4.97** is the official OID for organizationIdentifier. The contents of this field in the example certificate is the unique identifier for the legal person GoodAir.

Before issuing a certificate like the above, the TSP has to perform some validations to ensure that in the official source of truth (the business registry of Spain in this case) there is already registered information that states that the COO has indeed powers to act on behalf of GoodAir as a legal representative.

3.4.1.1.4 RealTruth: a TSP (Trust Service Provider)

RealTruth is an EU TSP, which appears in the TL (Trusted List) maintained by the [German Government](#).

RealTruth is a TSP based in Germany but operates in most of the countries of the EU and so being able to provide Trust Services across many countries, including Spain.

As all TSPs in the TLs of the Member States, its entry in the TL includes one or more "Services" entries which describe the Trust Services provided by the TSP.

A TSP can have one Service issuing certificates for signatures, another service issuing certificates for seals, another service for timestamping, etc.

The regulator approves or suspends each service from a TSP individually, and the services are the root anchor for a given trust environment.

In our case, the entry for RealTruth in the TL includes a **<TSPService>** entry, and inside it the **<ServiceDigitalIdentity>** entry includes a DER-encoded certificate specifying the digital identity of the root anchor for that trust domain. The certificate for the Service has in the Subject field:

2.5.4.97 = VATDE-170173453

CN = DRV QC 11 MA CA 2017ca

OU = QC 11 Mitarbeiter CA

O = Deutsche Rentenversicherung Westfalen

C = DE

Which in the **organizationIdentifier** field (OID 2.5.4.97) specifies the unique organisation identifier assigned by the German regulatory authority to the TSP: VATDE-170173453.



RealTruth provides **Legal Person Representative Certificates** which are qualified in accordance with Regulation (EU) No. 910/2014 of the European Parliament and of the Council of 23 July 2014 on electronic identification and trust services for electronic transactions in the internal market.

The certification policies of RealTruth includes the following sentences:

The Registration Authority must verify the following information in order to authenticate the identity of the organisation:

- The data concerning the business name or corporate name of the organisation.
- The data relating to the constitution and legal status of the subscriber.
- **The data concerning the extent and validity of the powers of representation of the applicant.**
- The data concerning the tax identification code of the organisation or equivalent code used in the country to whose legislation the subscriber is subject.

The TSP (RealTruth in this case) performs the verifications against **Authentic Sources**.

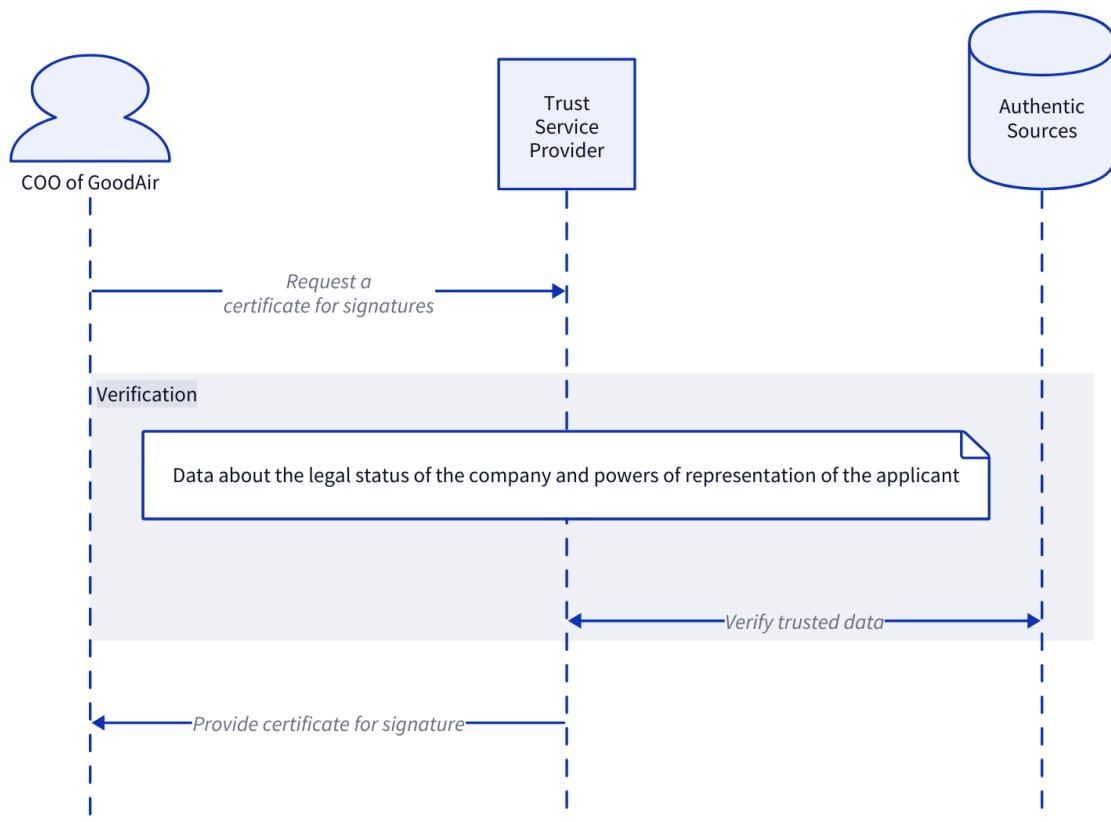
Authentic Sources are the public or private repositories or systems recognised or required by law containing attributes about a natural or legal person.

The Authentic Sources in scope of Annex VI of the [eIDAS2] legislative proposal are sources for attributes on address, age, gender, civil status, family composition, nationality, education and training qualifications titles and licences, professional qualifications titles and licences, public permits and licences, financial and company data.

Authentic Sources in scope of Annex VI are required to provide interfaces to Qualified Electronic Attestation of Attributes (QEAA) Providers to verify the authenticity of the above attributes, either directly or via designated intermediaries recognised at national level.

Authentic Sources may also issue (Q)EAA-s themselves if they meet the requirements of the eIDAS Regulation. It is up to the Member States to define terms and conditions for the provisioning of these services, but according to the minimum technical specifications, standards, and procedures applicable to the verification procedures for qualified electronic attestations of attributes.





In accordance to the policies, RealTruth made those validations when issuing the certificate to the COO, in such a way that the Relying Parties verifying the certificate can have a high level of trust in the assertion that the natural person identified in the certificate (with Spanish ID of 56565656V) is a legal representative of he GoodAir company (organizationIdentifier VATES-12345678).

3.4.1.1.5 John Doe: an administrative employee of GoodAir

This is an employee of the central administration department of GoodAir, who is going to be formally nominated to manage the onboarding process of GoodAir in the Data Space and some other operations in the Data Space once GoodAir is onboarded.. In particular, this employee is going to be nominated as a LEAR (Legal Entity Appointed Representative).

As its name implies, the LEAR has to be nominated by a **legal representative** of the GoodAir organisation with the necessary legal authority to commit the organisation for this type of decision. In our case, the COO of GoodAir will nominate John Doe as the LEAR of GoodAir, delegating to him the capabilities to perform onboarding in the SmartCityDS and some other associated tasks. That means that John Doe will not be empowered to perform other actions like onboarding on other Data Spaces or signing contracts or invoices on behalf of GoodAir.

To perform the nomination, the COO of GoodAir (a legal representative of GoodAir) will issue a special credential to John Doe and will sign the credential with his certificate for signatures as legal representative of GoodAir. The details are described later in this document.

3.4.1.2 Onboarding of organisations in DOME

This is an example of an onboarding process performed with the LEARCredential using the did:elsi method. The example is fully described in [Appendix I: example of onboarding of organisations in DOME](#).

3.4.2 Use of the DOME Trust and IAM framework by data/app service providers (Use case: Air Quality)

3.4.2.1 Description of roles of the Air Quality Monitoring app

Roles connected: admin, citizens

Depending on the role it allows doing something different. Example:

- Citizen can check temperature in the park, statistics of consumption of energy, park, etc.
- An admin can register additional sensors
- Registered sensors are the only ones able to inject measures

3.4.2.2 Acquisition of rights to use Air Quality Monitoring app by customer means that customer becomes Trusted issuer of that app specific VCs

The process of acquiring access to the air quality monitoring service is displayed. It is performed by employees of organisations that want to become trusted issuers for this service.

TODO: Sequence / Architecture Diagrams

1. The employee accesses DOME, in order to login.
2. The employee is displayed a list of Identity Providers for selecting the desired Identity Provider for login. The employee gets forwarded to a page for selecting the desired Identity Provider for login. One of the login options is “Verifiable Credentials” or something similar.
3. The employee selects the “Verifiable Credentials” login method, which causes the Marketplace portal to generate a QR containing the URL of the /authentication-requests endpoint of the Marketplace server.
4. The employee scans the QR with her mobile and the mobile calls the /authentication-requests endpoint.
5. This starts a standard SIOB (Self-Issued OpenID Provider) flow, where the Marketplace IDP plays the role of Relying Party (RP in Open ID Connect terminology) and the mobile device of the employee as a Self-Issued IDP. In this step, Marketplace IDP creates a SIOB Authentication Request. As a Self-Issued OP may be running locally as a native application or progressive web application (PWA), the RP may not have a network-addressable endpoint to communicate directly with the OP. We have to leverage the implicit flow of OpenID Connect to communicate with such locally-



running OPs, as described in https://openid.net/specs/openid-connect-self-issued-v2-1_0.html.

The Authentication Request travels in the response to the HTTP GET request performed in the previous point, as a JWT signed by the marketplace. The decoded contents of the JWT may be:

```
openid://?
  scope=openid
  &response_type=id_token
  &response_mode=post
  &client_id=did:elsi:EU.EORI.NLMARKETPLA
  &redirect_uri=https://marketplace.dome.org/siop_sessions
  &claims=...
  &registration={
    "subject_syntax_types_supported": ["did:key",
      "urn:ietf:params:oauth:jwk-thumbprint"]
  }
  &nonce=n-0S6_WzA2Mj
```

6. The Authentication Request is returned to the employee wallet acting as SIOP. The SIOP flow uses a new response mode **post** which is used to request the SIOP to deliver the result of the authentication process to a certain endpoint. The parameter **response_mode** is used to carry this value.

This endpoint where the SIOP shall deliver the authentication result is defined in the standard parameter **redirect_uri**.

7. In this step the employee verifies that the Marketplace is a trusted entity belonging to the ecosystem, by resolving the DID of the Marketplace which is received in the **client_id** parameter of the Authentication Request.

To resolve a DID, the wallet sends a GET request to the **/api/did/v1/identifiers/did:elsi:EU.EORI.NLMARKETPLA** endpoint of one of several trusted servers implementing the Universal Resolver functionality. The Universal Resolver includes a blockchain node, and there may be as many as needed. Its mission is to resolve DIDs using the blockchain and return the associated DID Document. The DID Document (as per W3C) contains relevant information about the entity owner of the DID. It contains its Public Key, used to verify the digital signature of the entity. It also contains the status of the entity in the Data Space ecosystem. It is extensible and can contain any public information which may be relevant for the use case. The Universal Resolver server must be operated by a trusted entity for the customer. There may be as many nodes as needed operated by different entities. At least one of those trusted entities has to be configured in the wallet of the employee.



8. The wallet receives the DID Document of Marketplace, with trusted information about the entity, including the Public Key associated with the Private Key that Marketplace uses to digitally sign tokens. For example:

```
{
  "payload": {
    "@context": [
      "https://www.w3.org/ns/did/v1",
      "https://w3id.org/security/v1"
    ],
    "id": "did:elsi:EU.EORI.NLMARKETPLA",
    "verificationMethod": [
      {
        "id": "did:elsi:EU.EORI.NLMARKETPLA#key-verification",
        "type": "JwsVerificationKey2020",
        "controller": "did:elsi:EU.EORI.NLMARKETPLA",
        "publicKeyJwk": {
          "kid": "key-verification",
          "kty": "EC",
          "crv": "secp256k1",
          "x": "V8XptJkb5wp1YkExcTF4nkyYVp7t5H5d5C4UPqCCM9c",
          "y": "kn3nSPxIIvd9iaG0N4v14ceuo8E4PcLXhhGeDzCE7VM"
        }
      }
    ],
    "service": [
      {
        "id": "did:elsi:EU.EORI.NLMARKETPLA#info",
        "type": "EntityCommercialInfo",
        "serviceEndpoint": "https://marketplace.dome.org/info",
        "name": "DOME"
      },
      {
        "id": "did:elsi:EU.EORI.NLMARKETPLA#sms",
        "type": "SecureMessagingService",
        "serviceEndpoint": "https://marketplace.fiware.io/api/sms"
      }
    ],
    "anchors": [
      {
        "id": "redt.alastria",
        "resolution": "UniversalResolver",
        "domain": "marketplace.dataspace",
        "ethereumAddress": "0xbcb9b29eeb28f36fd84f1CfF98C3F1887D831d78"
      }
    ],
    "created": "2021-11-14T13:02:37Z",
    "updated": "2021-11-14T13:02:37Z"
  }
}
```



9. The DID Document includes one or more public keys inside the “verificationMethod” array. The keys are identified by the “id” field in each element of the array. The employee wallet uses the **kid** field that was received in the Authentication Request (in the protected header of the JWT) to select the corresponding Public Key and verify the signature of the JWT. It also verifies that the top-level “**id**” field in the DID Document (“did:elsi:EU.EORI.NLMARKETPLA”) is equal to the **client_id** parameter of the Authentication Request.
10. The employee wallet creates an Authentication Response to be posted in the **redirect_uri** specified by Marketplace in step 5. The contents of the Authentication Response are described below.
11. The SIOP sends the authentication response to the endpoint passed in the **redirect_uri** authentication request parameter using a HTTP POST request using “application/x-www-form-urlencoded” encoding. The response contains an ID Token and a VP (Verifiable Presentation) token as defined in https://openid.net/specs/openid-connect-4-verifiable-presentations-1_0.html.

```
POST /siop_sessions HTTP/1.1
Host: marketplace.dome.org
Content-Type: application/x-www-form-urlencoded

id_token=eyJ0 ... NiJ9.eyJ1c ... I6IjlifX0.DeWt4Qu ... ZXso
&vp_token=...
&state=af0ifjsldkj
```

The decoded **id_token** would be:

```
{
  "iss": "https://self-issued.me/v2",
  "aud": "did:elsi:EU.EORI.NLMARKETPLA",
  "iat": 1615910538,
  "exp": 1615911138,
  "sub": "did:peer:99ab5bca41bb45b78d242a46f0157b7d",
  "auth_time": 1615910535,
  "nonce": "n-0S6_WzA2Mj",
}
```

The **sub** claim is *did:peer:99ab5bca41bb45b78d242a46f0157b7d* which is the DID of the user and for privacy reasons it is **not registered in any blockchain or centralised repository**. It must be the same as the DID included in the Verifiable Credential that was issued by the company/organisation when onboarding the employee and which travels in the authentication response.



The **vp_token** includes the Verifiable Presentation, which can be in two formats: **jwt_vp** (JWT encoded) or **ldp_vp** (JSON-LD encoded). The following example is using the JWT encoding:

```
{
  "format": "jwt_vp",
  "presentation": "eyJhbGciOiJSUzI1NilsInR5cCl6IkpxVCIsImtpZCI6ImRpZDpleGFtcGxIOmFiZmUxM2Y3MTIxMjA0

MzFjMjc2ZTEyZWNhYiNrZXIzLTEifQ.eyJzdWliOiJkaWQ6ZXhhbXBsZTpiYmZlYjFmNzEyZWJjNmYxY

zI3NmUxMmVjMjEiLCJqdGkiOiJodHRwOi8vZXhhbXBsZS5IZHUvY3JIZGVudGlhbHMvMzczMilsImIzc

yI6Imh0dHBzOi8vZXhhbXBsZS5jb20va2V5cy9mb28uandriiwibmJmljoxNTQxNDkzNzl0LCJpYXQiO

jE1NDE0OTM3MjQsImV4cCl6MTU3MzAyOTcyMywibm9uY2UiOiI2NjAhNjM0NUZTZXiLCJ2YI6eyJAY

29udGV4dCl6WyJodHRwczovL3d3dy53My5vcmcvMjAxOC9jcmVkJZGVudGlhbCJdLCJjcmVkJZGVudGlhbCJdLCJ2YI6eyJAY

3d3LnczLm9yZy8yMDE4L2NyZWRlbnRpYWxzL2V4YW1wbGVzL3YxIl0sInR5cGUIOlsvVmVyaWZpYWJsZ

UNyZWRlbnRpYWwiLCJVbml2ZXJzaXR5RGVncmVIQ3JIZGVudGlhbCJdLCJjcmVkJZGVudGlhbCJdLCJ2YI6eyJkZWdyZWUiOnsidHlwZSI6IkJhY2hlbG9yRGVncmVIIiwibmFtZSI6IjxzcGFulGxhbmc9J2ZyL

CI6eyJkZWdyZWUiOnsidHlwZSI6IkJhY2hlbG9yRGVncmVIIiwibmFtZSI6IjxzcGFulGxhbmc9J2ZyL

UNBJz5CYWNjYWxhdXLDqWF0IGVuIG11c2IxdWVzIG51bcOpclxdWVzPC9zcGFuPiJ9fX19.KLJo5GAy

BND3LDTn9H7FQokEsUEi8jKwXhGvoN3JtRa51xrNDgXD0cq1UTYB-rK4Ft9YVmR1NI_ZOF8oGc_7wAp

8PHbF2HaWodQIoOBxxT-4WNqAxft7ET6IkH-4S6Ux3rSGAmczMohEEf8eCeN-jC8WekdPl6zKZQj0YPB

1rx6X0-xIFBs7cl6Wt8rfBP_tZ9YgVWrQmUWypSioc0MUyiphmyEbLZagTyPIUyflGIEdqrZAv6eSe6RtxJy6M1-ID7a5HTzanYTWBPAUHDZGyGKXdJw-
```



```

W_x0IWChBzl8t3kpG253fg6V3tPgHeKXE94fz_QpYfg
--7kLsyBAfQGbg"
}

```

Which decoded could be:

```

{
  "@context": ["https://www.w3.org/2018/credentials/v1"],
  "type": ["VerifiablePresentation"],
  "verifiableCredential": [
    {
      "@context": [
        "https://www.w3.org/ns/credentials/v2",
        "https://happypets.fiware.io/2022/credentials/employee/v1"
      ],
      "id": "https://happypets.fiware.io/credentials/25159389-8dd17b796ac0",
      "type": ["VerifiableCredential", "EmployeeCredential"],
      "issuer": {
        "id": "did:elsi:EU.EORI.NLHAPPYPETS"
      },
      "issuanceDate": "2022-03-22T14:00:00Z",
      "validFrom": "2022-03-22T14:00:00Z",
      "expirationDate": "2023-03-22T14:00:00Z",
      "credentialSubject": {
        "id": "did:peer:99ab5bca41bb45b78d242a46f0157b7d",
        "verificationMethod": [
          {
            "id": "did:peer:99ab5bca41bb45b78d242a46f0157b7d#key1",
            "type": "JwsVerificationKey2020",
            "controller": "did:peer:99ab5bca41bb45b78d242a46f0157b7d",
            "publicKeyJwk": {
              "kid": "key1",
              "kty": "EC",
              "crv": "P-256",
              "x": "IJtvoA5_XptBvcfcrvtGCvXd9bLymmfBSSdNJf5mogo",
              "y": "fSc4gZX2R3QKKfHvS3m2vGSVSN8Xc04qsquyfEM55Z0"
            }
          }
        ],
        "roles": [
          {
            "target": "did:elsi:EU.EORI.NLPACKETDEL",

```



```

    "names": ["P.Create"]
  }
],
"name": "Jane Doe",
"given_name": "Jane",
"family_name": "Doe",
"preferred_username": "j.doe",
"email": "janedoe@packetdelivery.com"
}
}
]
}

```

12. Marketplace uses its own blockchain node or the one from a trusted entity implementing the Universal Resolver functionality to resolve the DID of the employee's company, which is inside the Verifiable Credential received in the Verifiable Presentation. This DID can be found in the "issuer" field of the "verifiableCredential" structure above.

Resolution is performed sending a GET request to the Universal Resolver: **/api/did/v1/identifiers/did:elsi:EU.EORI.NLHAPPYPETS**

Marketplace could use a Universal Resolver operated by a different entity, but this would reduce the level of trust compared to using its own server directly connected to the blockchain network.

13. Marketplace receives the DID Document of the employee's company with trusted information about the company, including the Public Key associated with the Private Key that the company used to digitally sign the Verifiable Credential that the employee has just sent inside a Verifiable Presentation as part of the authentication flow. **Using the Public Key and the DID inside the DID Document, it can verify the signature of the Verifiable Credential and that the company is a trusted entity in the ecosystem and that it is active.**
14. The above is just for verification of the Verifiable Credential. In addition, Marketplace can also verify that the Verifiable Presentation including the Verifiable Credential is sent by the employee and not by a malicious agent. To do so, it uses the Public Key of the employee in the "verificationMethod" of the "credentialSubject" structure. That public key is cryptographically bound to the employee DID during the onboarding process that the company performed with its employee.
15. Once all verifications have been performed, Marketplace creates an Access Token for the employee so she can use it to access services in the Marketplace server in the future.



16. The wallet (SIOP) receives the access token and saves it temporarily to be able to request services from Marketplace.
17. The wallet displays a success message to the employee.
18. The Marketplace server refreshes the page (it was the login page before) and displays the services available to the employee.

At this point the employee is logged in on DOME. The user is now able to use the services available to her.

At this moment, DOME knows the following:

- That the employee's company belongs to the Data Space and can issue credentials of the type EmployeeCredential because it is included in the corresponding Trusted Issuers Registry and is active, because this info is in the DID Document retrieved in step 13. Maintenance of this information is performed by the Trust Anchor entity (or entities) responsible for the Trusted Issuers Registry.
- That the employee's company says that the user is one of its employees. This info is inside the Verifiable Credential that is digitally signed by the company.

From this point on, DOME can display to the user the services available to her and execute them if the user is entitled to do so. DOME can use all the claims inside the credential to perform RBAC/ABAC access control and policy enforcement.

Now, when acquiring access to the air quality monitoring service, DOME will create the necessary access policies (or roles) at the Authorization Registry of the service provider, stating that the employee's company is allowed to issue credentials of the TODO type. The Authorization Registry implements Policy Administration Point (PAP) and Policy Management Point (PMP) functionalities.

It is out of scope for this document to describe the actual policy language and how access to the registry is performed.

3.5.2.3 For No Cheaper, the process is exactly the same as for the acquisition process for Happy Pets, except that the entities involved are No Cheaper Ltd and its employees, and that the basic offering is acquired. We do not provide a detailed flow to avoid repetition.

3.4.2.3 Operations by Admin as well as End users and implication in access

The process of accessing the air quality monitoring service is explained. In general, in the end the process allows access to an NGSI-LD service (based on a FIWARE Context Broker) based on the given access rights. This can be, e.g., retrieving entities (the air quality data) of this context broker or registering new sensors.



The following gives a detailed description of the process of accessing the service, when using Verifiable Credentials.

1. The user accesses the air quality monitoring company portal or starts the Air Quality Monitoring company app in its smartphone, to login.
2. The user gets forwarded to a page for selecting the desired Identity Provider for login. One of the login options is “Verifiable Credentials” or something similar.
3. The user selects the “Verifiable Credentials” login method, which causes the air quality monitoring company portal to generate a QR containing inside the URL of the /authentication-requests endpoint of the Air quality monitoring company IDP.
4. The user scans the QR with her mobile and the mobile calls the /authentication-requests endpoint.
5. This starts a standard SIOP (Self-Issued OpenID Provider) flow, where the Air Quality Monitoring company IDP plays the role of Relying Party (RP in Open ID Connect terminology) and the mobile device of the user as a Self-Issued IDP. In this step, Air Quality Monitoring company IDP creates a SIOP Authentication Request. As a Self-Issued OP may be running locally as a native application or progressive web application (PWA), the RP may not have a network-addressable endpoint to communicate directly with the OP. We have to leverage the implicit flow of OpenID Connect to communicate with such locally-running Ops, as described in https://openid.net/specs/openid-connect-self-issued-v2-1_0.html.

The Authentication Request travels in the response to the HTTP GET request performed in the previous point, as a JWT signed by Packet Delivery company. The decoded contents of the JWT may be:

```
openid://?  
  response_type=id_token  
  &response_mode=post  
  &client_id=did:elsi:EU.EORI.NLPACKETDEL  
  &redirect_uri=https%3A%2F%2Fidp-pdc.fiware.io%2Fsio_sessions  
  &scope=openid%20profile  
  &state=af0ifjsldkj  
  &nonce=n-0S6_WzA2Mj  
  &registration=%7B%22subject_syntax_types_supported%22:%5B%22did%22%5D,  
    %22id_token_signing_alg_values_supported%22:%5B%22RS256%22%5D%7
```

6. The Authentication Request is returned to the user wallet acting as SIOP. The SIOP flow uses a new response mode **post** which is used to request the SIOP to deliver the



result of the authentication process to a certain endpoint. The parameter **response_mode** is used to carry this value.

This endpoint where the SIOP shall deliver the authentication result is defined in the standard parameter **redirect_uri**.

7. In this step the user verifies that the Air Quality Monitoring company is a trusted entity belonging to the ecosystem, by resolving the DID of the Air Quality Monitoring company which is received in the **client_id** parameter of the Authentication Request.

To resolve a DID, the wallet sends a GET request to the **/api/did/v1/identifiers/did:elsi:EU.EORI.NLPACKETDEL** endpoint of one of several trusted servers implementing the Universal Resolver functionality. The Universal Resolver includes a blockchain node, and there may be as many as needed. Its mission is to resolve DIDs using the blockchain and return the associated DID Document. The DID Document (as per W3C) contains relevant information about the entity owner of the DID. It contains its Public Key, used to verify the digital signature of the entity. It also contains the status of the entity in the Data Space ecosystem. It is extensible and can contain any public information which may be relevant for the use case. The Universal Resolver server must be operated by a trusted entity for the user. There may be as many nodes as needed operated by different entities. At least one of those trusted entities has to be configured in the wallet of the user.

8. The wallet receives the DID Document of Air Quality Monitoring company, with trusted information about the company, including the Public Key associated with the Private Key that Packet Delivery company uses to digitally sign tokens. For example:

```
{
  "payload": {
    "@context": [
      "https://www.w3.org/ns/did/v1",
      "https://w3id.org/security/v1"
    ],
    "id": "did:elsi:EU.EORI.NLPACKETDEL",
    "verificationMethod": [
      {
        "id": "did:elsi:EU.EORI.NLPACKETDEL#key-verification",
        "type": "JwsVerificationKey2020",
        "controller": "did:elsi:EU.EORI.NLPACKETDEL",
        "publicKeyJwk": {
          "kid": "key-verification",
          "kty": "EC",
          "crv": "secp256k1",
          "x": "V8XptJkb5wpIYkExcTF4nkyYVp7t5H5d5C4UPqCCM9c",
          "y": "kn3nSPxIlvd9iaG0N4v14ceuo8E4PcLXhhGeDzCE7VM"
        }
      }
    ]
  }
}
```



```

],
"service": [
{
  "id": "did:elsi:EU.EORI.NLPACKETDEL#info",
  "type": "EntityCommercialInfo",
  "serviceEndpoint": "https://packetdelivery.com/info",
  "name": "Packet Delivery co."
},
{
  "id": "did:elsi:EU.EORI.NLPACKETDEL#sms",
  "type": "SecureMessagingService",
  "serviceEndpoint": "https://packetdelivery.com/api"
}
],
"anchors": [
{
  "id": "redt.alastria",
  "resolution": "UniversalResolver",
  "domain": "packetdelivery.ala",
  "ethereumAddress": "0xbcB9b29eeb28f36fd84f1CfF98C3F1887D831d78"
}
],
"created": "2021-11-14T13:02:37Z",
"updated": "2021-11-14T13:02:37Z"
}
}
}

```

9. The DID Document includes one or more public keys inside the “verificationMethod” array. The keys are identified by the “id” field in each element of the array. The user wallet uses the kid field that was received in the Authentication Request (in the protected header of the JWT) to select the corresponding Public Key and verify the signature of the JWT. It also verifies that the top-level “id” field in the DID Document (“did:elsi:EU.EORI.NLPACKETDEL”) is equal to the **client_id** parameter of the Authentication Request.
10. The user wallet creates an Authentication Response to be posted in the **redirect_uri** specified by Air Quality Monitoring company in step 5. The contents of the Authentication Response are described below.
11. The SIOP sends the authentication response to the endpoint passed in the **redirect_uri** authentication request parameter using a HTTP POST request using “application/x-www-form-urlencoded” encoding. The response contains an ID Token and a VP (Verifiable Presentation) token as defined in https://openid.net/specs/openid-connect-4-verifiable-presentations-1_0.html.



```
POST /siop_sessions HTTP/1.1
Host: client.example.com
Content-Type: application/x-www-form-urlencoded

id_token=eyJ0 ... NiJ9.eyJ1c ... l6ljlfX0.DeWt4Qu ... ZXso
&vp_token=...
&state=af0ifjsldkj
```

The decoded **id_token** would be:

```
{  
  "iss": "https://self-issued.me/v2",  
  "aud": "did:elsi:EU.EORI.NLPACKETDEL",  
  "iat": 1615910538,  
  "exp": 1615911138,  
  "sub": "did:peer:99ab5bca41bb45b78d242a46f0157b7d",  
  "auth_time": 1615910535,  
  "nonce": "n-0S6_WzA2Mj"  
}
```

The **sub** claim is `did:peer:99ab5bca41bb45b78d242a46f0157b7d` which is the DID of the user and that is not registered in any blockchain or centralised repository. It must be the same as the DID included in the VP that was issued by the user's company when onboarding her and which travels in the authentication response.

The **vp_token** includes the Verifiable Presentation, which can be in two formats: **jwt_vp** (JWT encoded) or **ldp_vp** (JSON-LD encoded). The following example is using the JWT encoding:

```
{  
  "format": "jwt_vp",  
  "presentation":  
    "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6ImRpZDpleGFtcGxIOMFiZmUxM2Y3MTIxMjA0  
    MzFjMjc2ZTEyZWNhYiNrZXlZTLEifQ.eyJzdWlOiJkaWQ6ZXhhbXBsZTpIYmZIYjFmNzEyZWJjNmYxY  
    zI3NmUxMmVjMjEiLCJqdGkiOjodHRwOi8vZXhhbXBsZS5IZHUvY3JIZGVudGlhbHMvMzczzMilsImlc  
    yI6Imh0dHBzOi8vZXhhbXBsZS5jb20va2V5cy9mb28uandriiwibmJmljoxNTQxNDkzNzI0LCJpYXQiO  
    jE1NDE0OTM3MjQsImV4cCI6MTU3MzAyOTcyMywibm9uY2UiOii2NjAhNjM0NUTZXliLCJ2YyI6eyJAY  
    29udGV4dCI6WyJodHRwczovL3d3dy53My5vcmcvMjAxOC9jcmVkJZw50aWFscy92MSIsImh0dHBzOi8vd  
    3d3LnczLm9yZy8yMDE4L2NyZWRlbnRpYWxzL2V4YW1wbGVzL3YxIl0sInR5cGUIolsiVmVyaWZpYWJsZ  
    UNyZWRlbnRpYWwiLCJVbml2ZXJzaXR5RGVncmVIQ3JIZGVudGlhbCjdLCJcmVkJZw50aWFsU3ViamVjd  
    CI6eyJkZWdyZWUiOnsidHlwZSI6IkJhY2hlbG9yRGVncmVIIiwibmFtZSI6ljzcGFulGxhbmc9J2ZyL  
    UNBjz5CYWNjYWxhdXLDqWF0IGVuIG11c2lxWVzIG51bcOpclmxWVzPC9zcGFuPiJ9fX19.KLJo5GAy  
    BND3LDtN9H7FQokEsUEi8jKwXhGvoN3JtRa51xrNDgXDb0cq1UTYB-rK4Ft9YVmR1NI_ZOF8oGc_7wAp  
    8PPhBF2HaWodQloOBxxT-4WNqAxft7ET6IkH-4S6Ux3rSGAmczMohEEf8eCeN-iC8WekdPI6zKZQj0YPB
```

```

1rx6X0-xIFBs7cl6Wt8rfBP_tZ9YgVWrQmUWypSioc0MUyiphmyEbLZagTyPIUyflGIEdqrZAv6eSe6R
txJy6M1-ID7a5HTzanYTWBPAUHDZGyGKXdJw-W_x0IWChBzl8t3kpG253fg6V3tPgHeKXE94fz_QpYfg
--7kLsyBAfQGbg"
}

```

Which decoded could be:

```

{
  "@context": ["https://www.w3.org/2018/credentials/v1"],
  "type": ["VerifiablePresentation"],
  "verifiableCredential": [
    {
      "@context": [
        "https://www.w3.org/ns/credentials/v2",
        "https://happypets.fiware.io/2022/credentials/employee/v1"
      ],
      "id": "https://happypets.fiware.io/credentials/25159389-8dd17b796ac0",
      "type": ["VerifiableCredential", "CustomerCredential"],
      "issuer": {
        "id": "did:elsi:EU.EORI.NLHAPPYPETS"
      },
      "issuanceDate": "2022-03-22T14:00:00Z",
      "validFrom": "2022-03-22T14:00:00Z",
      "expirationDate": "2023-03-22T14:00:00Z",
      "credentialSubject": {
        "id": "did:peer:99ab5bca41bb45b78d242a46f0157b7d",
        "verificationMethod": [
          {
            "id": "did:peer:99ab5bca41bb45b78d242a46f0157b7d#key1",
            "type": "JwsVerificationKey2020",
            "controller": "did:peer:99ab5bca41bb45b78d242a46f0157b7d",
            "publicKeyJwk": {
              "kid": "key1",
              "kty": "EC",
              "crv": "P-256",
              "x": "IJtvoA5_XptBvcfcrvtGCvXd9bLymmfBSSdNJf5mogo",
              "y": "fSc4gZX2R3QKKfHvS3m2vGSVSN8Xc04qsquyfEM55Z0"
            }
          }
        ],
        "roles": [
        ]
      }
    }
  ]
}

```



```

    "target": "did:elsi:EU.EORI.NLPACKETDEL",
    "names": ["P.Info.gold"] // Or P.Info.standard
  },
],
"name": "Jane Doe",
"given_name": "Jane",
"family_name": "Doe",
"preferred_username": "j.doe",
"email": "janedoe@packetdelivery.com"
}
}
]
}

```

12. Air Quality Monitoring company uses its own blockchain node implementing the Universal Resolver functionality to resolve the DID of the user's company, which is inside the Verifiable Credential received in the Verifiable Presentation. This DID can be found in the "issuer" field of the "verifiableCredential" structure above.
 Resolution is performed sending a GET request to the Universal Resolver:
/api/did/v1/identifiers/did:elsi:EU.EORI.NLHAPPYPETS
 Air Quality Monitoring could use a Universal Resolver operated by a different entity, but this would reduce the level of trust compared to using its own server directly connected to the blockchain network.
13. Air Quality Monitoring receives the DID Document of the user's company with trusted information about the company, including the Public Key associated with the Private Key that the company used to digitally sign the Verifiable Credential that the user has just sent inside a Verifiable Presentation as part of the authentication flow. **Using the Public Key and the DID inside the DID Document, it can verify the signature of the Verifiable Credential and that the user's company is a trusted entity in the ecosystem.**
14. The above is just for verification of the Verifiable Credential. In addition, the Air Quality Monitoring company can also verify that the Verifiable Presentation including the Verifiable Credential is sent by the user and not by a malicious agent. To do so, it uses the Public Key of the user in the "verificationMethod" of the "credentialSubject" structure. That public key is cryptographically bound to the user DID during the onboarding process that the company performed with the user.
15. Once all verifications have been performed, Air Quality Monitoring company creates an Access Token for the user so she can use it to access services in the Air Quality Monitoring company in the future.
16. The wallet (SIOP) receives a successful reply to the POST request.



17. The Air Quality Monitoring company proxy notifies the Air Quality Monitoring portal that the user is successfully authenticated, and the portal can display the services available to that user. The browser of the user receives the Access Token created by Air Quality Monitoring to enable it to request services without going through the previous authentication process. The Access Token is a standard OAuth access token that includes the information that Air Quality Monitoring requires for accessing its services.

At this point the user is logged in on the Air Quality Monitoring company portal/app and is presented with the possible services provided, e.g., to retrieve current air quality data.

At this moment, the Air Quality Monitoring company knows the following:

- The user's company is a participant in the Data Space and that it is a Trusted Issuer of EmployeeCredentials because this info is in the DID Document retrieved in step 13. Maintenance of this information is performed by the Trusted Anchor entity(or entities) managing the Trusted Participants List and Trusted Issuers Registry.
- The user's company says that the user is a customer. This info is inside the Verifiable Credential that is digitally signed by the company.
- The category of the user (and associated policies) with regards to the services offered by the Air Quality Monitoring company. This information is also in the Verifiable Credential presented by the user.

18. The user is presented with the possible services provided by Air Quality Monitoring.
19. The user requests to retrieve the current temperature in the park in the Air Quality Monitoring portal/app.
20. Air Quality Monitoring company portal/app sends a request to Air Quality Monitoring proxy (API gateway + PEP), in order to get the temperature measurement entity in the park. The request contains the Access Token generated in step 15, with information about the authorisation registry to retrieve policies from.

```
> Authorization: Bearer IleD...NIQ // Bearer JWT
```

```
> Content-Type: application/json
```

```
PATCH https://umbrella.fiware.io/ngsi-ld/v1/entities/urn:ngsi-  
Id:DELIVERYORDER:001/attrs/pta
```

```
> Payload
```

```
{  
  "value": "<new PTA>",  
  "type": "Property"  
}
```

```
Decoded Bearer JWT payload:
```

```
{
```



```

"iss": "EU.EORI.NLHAPPYPETS", // Issuer: Happy Pets
"sub": "419404e1-07ce-4d80-9e8a-eca94vde0003de", // User pseudonym
"jti": "d8a7fd7465754a4a9117ee28f5b7fb60",
"iat": 1591966224,
"exp": 1591966254,
"aud": "EU.EORI.NLHAPPYPETS",
"authorisationRegistry": { // AR to retrieve policies from
    "url": "https://ar.packetdelivery.com",
    "identifier": "EU.EORI.NLHAPPYPETS",
    "delegation_endpoint": "https://ar.packetdelivery.com/delegation",
}
}

```

21. Air Quality Monitoring company proxy received the request of step 19 for reading the current temperature measurement entity at the park. The Access Token received from the user ensures that she was assigned the delegation evidence with a policy for reading the temperature entity for this specific location (called issuance at **user level**). Furthermore, since in this scenario the required user policy was issued by a 3rd party (the user's company), the proxy has to check whether the user's company itself is allowed to delegate this policy. In general, the rule would be that the proxy needs to check the existence of valid policies through the chain of issuers, until itself (in this case the Air Quality Monitoring company) is the issuer. In this scenario, the proxy will check policies at two different levels: issued at **organisational level** (from Air Quality Monitoring company to the user's company) and issued at **user level** (from the user's company to the user). The Verifiable Credential takes care of the user level policies.

At first, the Air Quality Monitoring company proxy validates the JWT which is part of the authorization header of the PATCH request.

22. In order to check whether the user's company is allowed to delegate the policy to its users, the proxy will check at the Air Quality Monitoring company Authorisation Registry whether this policy exists. The proxy sends a request to the /delegation endpoint of the Air Quality Monitoring company Authorization Registry.
23. The proxy receives the delegation evidence policy issued from Air Quality Monitoring company to the user's company.
24. Having received the delegation information from the Air Quality Monitoring company Authorization Registry, the proxy (or more precisely, the PDP) can now evaluate whether the contained **organisational policy** allows for reading the temperatur, and therefore whether the user's company is allowed to delegate the access to its users. If the proxy received a valid policy, access would be granted on an **organisational level**.



If the requested delegation evidence can not be found or the returned policy contains the Deny rule, the request for reading the temperature would be denied by the Air Quality Monitoring company proxy and an error would be returned to the Air Quality Monitoring company portal/app, also presented to the user. The following steps would be omitted.

25. As described in the previous steps, the PDP evaluated that reading the temperature of a temperature measurement entity at a specific location is granted, both on **organisational level** and **user level**. As a result, the request for reading the temperature is forwarded by the Air Quality Monitoring company proxy to the Air Quality Monitoring company Context Broker which holds the information of the temperature measurement entities. The Context Broker returns a successful response containing the requested information. The Context Broker response is returned to the Air Quality Monitoring company portal, in response to the request of step 26.
26. The temperature value is presented to the user.

3.4.3 Use of the DOME Trust and IAM framework by data/app service providers (Use Case: Packet Delivery)

The use case implements a scenario where a data service provider offers a service on DOME, so that service consuming parties can acquire access to this offering. Furthermore, these consuming parties can delegate the access to the acquired service offering to their users (e.g., customers).

In this use case, the provider is a packet delivery company, supporting creation and management of packet delivery orders and offering a service to view and change specific attributes of a packet delivery order. The consuming parties will be different retailers providing shop systems to their customers. These retailers will acquire access to services of the packet delivery company through DOME, and delegate the access to its customers.

In the use case, several parties are involved, each hosting its own infrastructure. Namely:

- Marketplace: DOME as public marketplace for creating service offerings and acquiring access to them
- Trust Anchor: fulfils the role of a scheme administrator which holds information about each participating party (including a global UID called EORI) and allows it to check for the admittance of each party.
- Packet Delivery Company: Provider which offers a service for retrieving and updating data of packet delivery orders



- Happy Pets: Premium pets retailer. Additionally there are two human actors involved: Happy Pets employee (actor working on behalf of Happy Pets company) and Happy Pets Customer (Customer of the pets shop system)
- No Cheaper: Retailer offering products at big discounts. Additionally there are two human actors: No Cheaper employee (actor working on behalf of No Cheaper company) and No Cheaper Customer (Customer of the No Cheaper shop system)

The following figure depicts the overall architecture of the reference use case. The packet delivery company and the shop system provider each have their own identity provider and authorization registry. In addition, the packet delivery company hosts a portal which allows users to view and modify attributes of packet delivery orders. The order entities are stored in an instance of the Context Broker. Read and write access to the packet delivery order entities is controlled by a PEP Proxy and PDP according to the described roles in section 3.4.3.1 Parties involved.

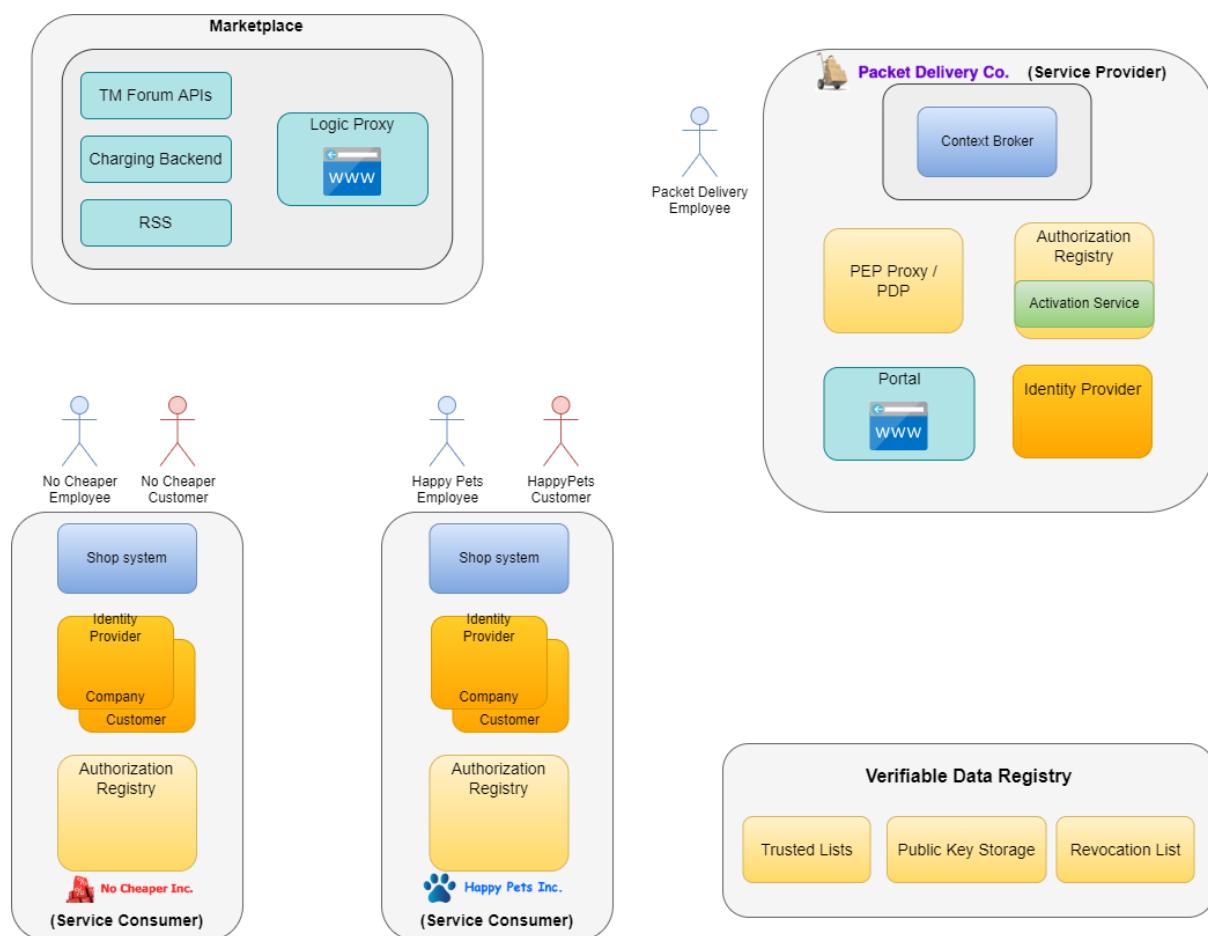


Figure 35 - Overall architecture

3.4.3.1 Parties involved



3.4.3.1.1 Data Service Provider: Packet Delivery Company

Packet Delivery Company (PacketDelivery for short) is a parcel service provider delivering packets all over the world. It offers two kind of Packet Delivery services:

- A “Standard Packet Delivery” service for which the customer simply is given the opportunity to specify the issuer (sender) of the packet, the address, date and time at which the packet to be delivered is ready for collection, and the name and address of the destinee to whom the package has to be delivered. When the PacketDelivery receives a packet delivery order from a given customer, it returns the target date at which the packet is planned to be delivered. Under defined terms and conditions (e.g., there are no problems with customs, addresses are valid, etc), it commits to deliver the packet in 48 hours within the same country and 5-6 days if it requires international shipping. However, customers are not allowed to adjust the concrete date of delivery (e.g., delaying it to a more suitable date) nor fine-tune the concrete time of delivery within the selected date of delivery.
- A “Gold Packet Delivery” service for which the customer enjoys all the benefits of the “Standard Packet Delivery” but also is allowed to adjust the concrete address of delivery, date of delivery within an offered period, as well as concrete time of delivery within the selected date of delivery, provided such adjustments are feasible (e.g., are requested enough time in advance and do not imply additional costs).

PacketDelivery offers its services electronically to different retailers, bringing them access to its Packet Delivery Info system (P.Info) via a REST API in order to allow them to issue packet delivery orders, trace location of orders and allow their customers to perform requests for adjustments on address, date and time of planned delivery when their clients are entitled to.

This is implemented because the P.Info system offers access to data about DELIVERYORDER entities through a Context Broker using NGSI-LD. A DELIVERYORDER is an entity with attributes like:

- issuer
- pickingAddress
- pickingDate
- pickingTime
- destinee
- deliveryAddress
- PDA (planned date of arrival)
- PTA (planned time of arrival)
- EDA (expected date of arrival)
- ETA (expected time of arrival)

PacketDelivery has defined two roles “P.Info.standard” and “P.Info.gold” for the P.Info system based on which the operations that can be requested on the above attributes through the Context Broker service it publishes have been defined. To simplify the description of the scenario, we will focus on attributes *deliveryAddress*, *PDA* and *PTA* since we could assume that the other ones will be assigned values at the time an order is created, will be always readable but will not be able to be changed by users with the defined roles. In that sense, the following policies apply for the defined roles regarding modification of these three attributes (*deliveryAddress*, *PDA*, *PTA*) once an order has been created:



Path:	Verb	
/ngsi- Id/v1/entities/{entityID}/ attrs/{attrName}	GET	PATCH
deliveryAddress	P.Info.standard/gold	P.Info.gold
EDA	P.Info.standard/gold	---
ETA	P.Info.standard/gold	---
PDA	P.Info.standard/gold	P.Info.gold
PTA	P.Info.standard/gold	P.Info.gold

Note that orders will be created using POST but with a different path (/ngsi-**Id**/v1/entities/). For issuing such requests an additional role “P.Create” is defined which will be assigned to the retailers Happy Pets and No Cheaper only.

PacketDelivery has decided to publish two different Packet Delivery offerings targeted to potential retailers and other kind of companies:

- Basic Delivery: which allows the company which acquires the offer to provide just a Standard Packet Delivery Services to its customers
- Premium Delivery: which allows the company which acquires the offer to provide Standard and Gold Delivery Services to its customers

Both have different pricing assigned.

Note that PacketDelivery should not know about the identity of users of applications of any Retailer company. It simply should be able, when it receives a request, to a) recognize that such request comes from a user linked to an application that belongs to a Retailer company that acquired one of its offerings in the Marketplace, b) find out what is the role within the P.Info application that such user has been assigned by the given Retailer company (i.e., either “P.Info.standard” or “P.Info.gold”), and c) check that such a role is a role that the given Retailer company could assign, considering the offering in the Marketplace it had acquired. After such steps, PacketDelivery will simply check whether a user with the given role can perform the operation requested.

An application created by organization NoCheaper, no matter if it defines users whom it assigns role “P.Info.gold” to, is unable to successfully change the value of the PTA attribute of a given order because it has acquired the Standard Packet Delivery service which does not allow to change those values.



3.4.3.1.2 Data Service Consumer: Happy Pets Inc.

HappyPets Inc. (HappyPets for short) is a company that sells products for pets. It will acquire the “Premium Packet Delivery” offering on DOME. This will allow it to offer, in turn, standard and gold delivery services to its customers through the store application of HappyPets (HappyPetsStore). In addition, there may be certain employees within its own organisation, namely supervisors and agents in the phone help-desk service it offers, who may change the deliveryAddress, PDA and PTA of a given order using an internal application (HappyPetsBackOffice).

When a customer signs up in the HappyPetsStore, it can act as “regular” customer or “prime” customer (paying an annual fee). “Regular” customers are provided the standard packet delivery services while “prime” customers are provided the gold packet delivery service. This means they are assigned the “P.Info.standard” role and the “P.Info.gold” role within the HappyPetsStore application, respectively.

On the other hand, different employees are given different roles within the HappyPetsBackOffice application, so certain employees with supervisor roles at physical shops or agents at the central help-desk also have the “P.Info.gold” role assigned.

The Happy Pets employee:

- Acquires the offering “Premium Packet Delivery” at DOME

The Happy Pets Customer:

- Signs up at the shop system of Happy Pets and gets assigned the “prime customer” role
 - For simplicity, we will assume that there is already a Happy Pets customer which already registered as “prime customer”
- Makes an order on the shop system, which results in the creation of a packet delivery order
 - For simplicity, we will assume that there is already a delivery order for this customer at the Packet Delivery company system
- Successfully changes the PTA of the order via the packet delivery company portal
 - We will describe later in this document the detailed process to perform this operation

3.4.3.1.3 Data Service Consumer: No Cheaper Ltd.

NoCheaper Ltd (NoCheaper for short) is a company that sells products of any kind at rather big discounts. It will acquire the “Basic Packet Delivery” offering from the Packet Delivery Service company in the Marketplace.

The No Cheaper employee:

- Acquires the offering “Basic Packet Delivery” at DOME



The No Cheaper Customer:

- Signs up at the No Cheaper shop system and gets assigned the “standard customer” role
 - For simplicity, we will assume that there is already a No Cheaper customer which already registered as “standard customer”
- Makes an order on the shop system, which results in the creation of a packet delivery order
 - For simplicity, we will assume that there is already a delivery order for this customer at the Packet Delivery company system
- When trying to change the PTA of the order via the packet delivery company portal, it is denied
- It can be also shown that this request will get denied, even when the No Cheaper employee is assigning the “Prime Customer” role to the No Cheaper customer in its own Identity Provider system

3.4.3.1.4 Trust Anchor Framework

The system uses Verifiable Credentials and participants are identified via DIDs (described in “[Decentralized Identifiers \(DIDs\) v1.0](#)”). In order to enable an efficient and decentralised verification of the credentials and identities of participants, a blockchain-based Trust Framework has to be implemented to avoid central entities intermediation in all authentication flows.

The trust framework is basically composed of two things:

1. A list of the identities of trusted organisations stored in the blockchain, together with associated information for each entity.
2. A process to add, modify and delete the trusted entities, implementing a concrete governance model.

The trust framework is designed to be largely decentralised and represents the trust relationships in the real world. Here we describe a possible approach to implementing a blockchain-based trust framework which is very decentralised and at the same time simple, secure and robust.

The identities of the legal persons involved in the ecosystem are registered in a common directory implemented in the blockchain following a hierarchical scheme very similar to the DNS (Domain Name Service) schema in the Internet.

Essentially, once an entity is registered in the system, it is completely autonomous for adding other entities that are managed as child entities.

In this way, trust is delegated according to a well defined, transparent, auditable and public scheme. Any participant can get trusted information about the current trust structure of the ecosystem and also the events that led the system to the current situation. For example, what entity registered another entity, when it was done and what attributes were assigned to the child entity by the parent entity.

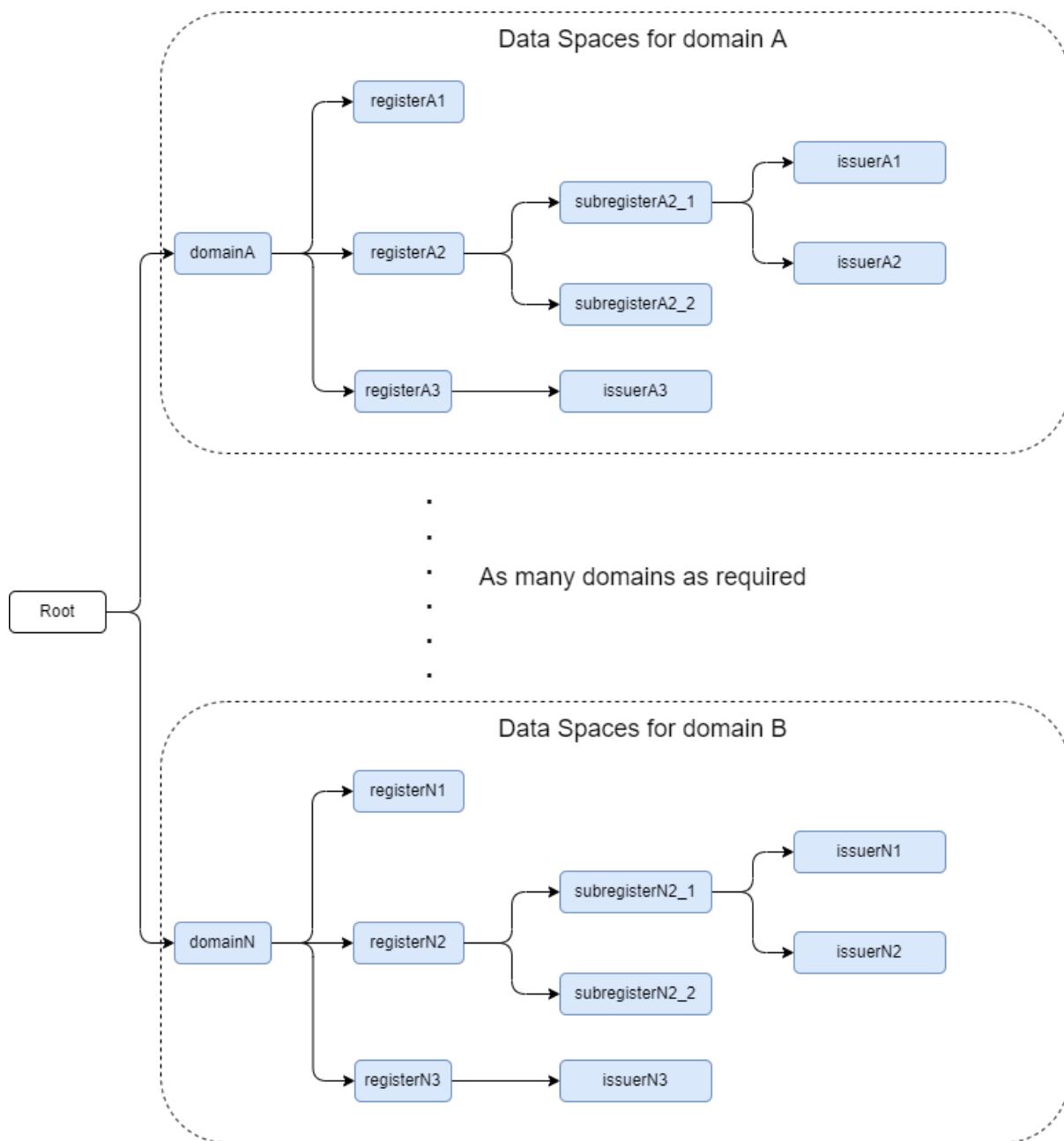


The scheme is flexible enough to implement as many levels as required by the actual governance model of the ecosystem. In very simple cases, it can have just two levels, where there is only one entity registering all other participants in the ecosystem.

In general, the system can be made very decentralised. However, there is one centralised element: the root of trust at the top of the hierarchy should be a trusted entity (or federation of entities) in the ecosystem that is the one responsible for bootstrapping the system. Depending on the concrete governance framework of the ecosystem, this may be the only mission of the root entity, possibly including the monitoring and oversight of the ecosystem. Typically it should be a regulatory body, a public administration or a neutral organisation which is accepted as fully trusted by all participants in the ecosystem.

The approach for a single blockchain network is described in the following figure (the scheme is easily extensible to different blockchain networks where we want to establish trust across them so entities in one network can interact with entities in another network in a trusted way).





As can be seen in the figure, the Trust Framework in a given blockchain is not really a flat list, but a hierarchical structure, implemented as a Smart Contract:

- There is a special organisation (or set of organisations) which is at the root of the hierarchy. This entity is called the Trusted Registration Authority (TRA) in EBSI, or Trusted Anchors in other contexts. We will use the term Trusted Anchors in the following description. The essential characteristic is that this is the most trusted entity/entities in the ecosystem.
- This root entity is responsible for registering the identities of some other trusted entities. For example, in a country with several regions with autonomous competencies to manage universities, the Ministry of Education could register in the blockchain the

identities of the regional institutions which are responsible for managing the universities in each of their regions.

- Once this is done, each of the regional institutions can register the identities of dependent entities, like universities.
- The hierarchy can have several levels. For example, a university can be big and have several organisational units with some autonomy, maybe distributed geographically. It can create sub-identities and register them as child nodes in the blockchain.

3.4.3.2 Verifiable Credentials in the ecosystem

In this section we describe the different types of credentials that are needed for the functionalities in the ecosystem.

3.4.3.2.1 Employee of Packet Delivery

Packet Delivery issues credentials to some of its employees, so they can access DOME, either to create offerings or to purchase offerings.

This credential is used by an employee of Packet Delivery to prove to a third party that she is entitled to use some services provided by the third party on behalf of the employed company (Packet Delivery in this case). In other words, the credential is used as a mechanism for Packet Delivery to delegate its access control rights to one or more of its employees.

The essential characteristics of such credential are:

- Nobody has tampered with its contents since it was issued, because the credential is digitally signed by the issuer, Packet Delivery.
- Proves that the issuer is Packet Delivery, because the public key that verifies the signature of the credential is cryptographically associated with the real-world identity of Packet Delivery registered in the Trust Framework.
- Optionally, it can prove that the credential was issued no later than a given time, because the credential was registered (timestamped) in the blockchain when it was issued. The term “notarisation” is commonly used for this action, but it is wrong, because the term is coming from anglo-saxon cultures where notaries are very different from the latin-germanic notary functions in the EU and many other countries in the world. We will use the term “timestamping”.

Please note that the date of timestamping can be greater than the date in the field “Issued at” included inside the credential. For example, the credential is created and signed at one time, but timestamped the next day (maybe to batch the operation with other credentials). The real requirement is that nobody can create a credential and timestamp as if it happened in the past. In other words, nobody can create credentials from the past. The verifiers have to check that the field inside the credential “Issued at” is not later than the timestamp (at least by a small leeway to account for clock synchronisation differences).

Also note that many credentials may not require timestamping, avoiding the overhead of the registration process. It all depends on the type of credential, the intended usage of the



credential and the level of risk assumed. The employee credential discussed here is one example of credential that does not require timestamping with the same level of risk. The only thing that the verifier requires is that the holder can prove that at the time of usage of the credential (eg., login), the credential was issued by the employer (Packet Delivery in our case). Obviously, this does not require timestamping, because if the employee can present a credential when performing login, she can do so only if the credential was issued before.

From the above description we can derive the following trust properties for a verifier receiving a credential:

- The level of trust in the identity of the issuer of the credential depends on the level of trust of the verifier in the onboarding process implemented in the Trust Framework. The onboarding process associates the public key of the issuer with its real identity.
- The level of trust in the claims inside the credential depends on the level of trust that the verifier has with the issuer entity. For example, Packet Delivery could issue employee credentials to people who are not real employees. However, if this is the case the verifier has a strong non-repudiable mechanism to prove to third-parties (e.g., a court) that the issuer stated wrong facts.

From the above it follows that Packet Delivery can issue employee credentials which include some employee data (name, surname, etc.) and the verifier can have a given level of trust on those claims.

But this just proves that Packet Delivery attests that the data inside the credential (called claims) is true. It does not say anything about whether the person presenting the credential online is the same that is referred to in the claims. In other words, the person sending the employee credential to the verifier could be a different person from the employee.

This is the reason why the credential includes a public key as one of the claims associated with the employee (inside the “`credentialSubject`” object).

That public key corresponds to a private key that was generated in the employee device (PC or mobile) during the process of credential issuance. The process is explained in more detail later, but essentially:

- The employee generates a pair of public/private keys and sends the public key to the employer via an authenticated and encrypted channel (e.g., HTTPS). This channel can be the usual mechanism that employees use to connect to enterprise applications.
- The employer generates a credential with some employee data and includes the public key.
- The employer signs the credential and sends it to the employee using the same authenticated channel.

Below we present an example employee credential issued by Packet Delivery.



```

// Credential issued by PacketDelivery to its employees, providing access to
// Marketplace, either to create offerings or to purchase offerings.
{
  "@context": [
    "https://www.w3.org/ns/credentials/v2",
    "https://marketplace.fiware.io/2022/credentials/employee/v1"
  ],
  "id": "https://pdc.fiware.io/credentials/6e14b8b8-87fa0014fe2a",
  "type": ["VerifiableCredential", "EmployeeCredential"],
  "issuer": {
    "id": "did:elsi:EU.EORI.NLPACKETDEL"
  },
  "issuanceDate": "2022-03-22T14:00:00Z",
  "validFrom": "2022-03-22T14:00:00Z",
  "expirationDate": "2023-03-22T14:00:00Z",
  "credentialSubject": {
    "id": "did:peer:99ab5bca41bb45b78d242a46f0157b7d",
    "verificationMethod": [
      {
        "id": "did:peer:99ab5bca41bb45b78d242a46f0157b7d#key1",
        "type": "JwsVerificationKey2020",
        "controller": "did:peer:99ab5bca41bb45b78d242a46f0157b7d",
        "publicKeyJwk": {
          "kid": "key1",
          "kty": "EC",
          "crv": "P-256",
          "x": "IJtvoA5_XptBvcfcrtGCvXd9bLymmfBSSdNJf5mogo",
          "y": "fSc4gZX2R3QKKfHvS3m2vGSVSN8Xc04qsquyfEM55Z0"
        }
      }
    ],
    "roles": [
      {
        "target": "did:elsi:EU.EORI.NLMARKETPLA",
        "names": ["seller", "buyer"]
      }
    ],
    "name": "Jane Doe",
    "given_name": "Jane",
    "family_name": "Doe",
    "preferred_username": "j.doe",
    "email": "janedoe@packetdelivery.com"
  }
}

```



```
}
```

The structure of the above credential can be visualized as follows:

EmployeeCredential			
@context	https://www.w3.org/2018/credentials/v1		
	https://marketplace.fiware.io/2022/credentials/employee/v1		
id	https://pdc.fiware.io/credentials/6e14b8b8-87fa0014fe2a		
type	VerifiableCredential EmployeeCredential		
issuer	id did:elsi:EU.EORI.NLPACKETDEL		
issuanceDate	2022-03-22T14:00:00Z		
validFrom	2022-03-22T14:00:00Z		
expirationDate	2023-03-22T14:00:00Z		
credentialSubject	id	did:peer:99ab5bca41bb45b78d242a46f0157b7d	
	verificationMethod	id	did:peer:99ab5bca41bb45b78d242a46f0157b7d#key1
		type	JwsVerificationKey2020
		controller	did:peer:99ab5bca41bb45b78d242a46f0157b7d
		publicKeyJwk	kid key1
			kty EC
			crv P-256
			x IJtvoA5_XptBvcfcrvtGCvXd9bLymmfBSSdNJf5mogo
			y fSc4gZX2R3QKKfHvS3m2vGSVSN8Xc04qsquyfEM55Z0
	roles	target	did:elsi:EU.EORI.NLMARKETPLA
		names	seller
			buyer
	name	Jane Doe	
	given_name	Jane	
	family_name	Doe	
	preferred_username	j.doe	
	email	janedoe@packetdelivery.com	

The credential is of type “*EmployeeCredential*” and to enable access to DOME the roles embedded in it can be “buyer”, “seller” or both. The URL in the “@context” field points to the marketplace (<https://pdc.fiware.io/2022/credentials/employee/v1>), which defines the general requirements for an Employee Credential. However, participants in the ecosystem can extend it and of course use the roles and role names that they need for their own purposes.

The “*credentialSubject*” section in the credential has the following objects:

- “**id**”, specified as a DID. For privacy reasons and given that this is a natural person, the DID used is the Peer Method as specified in the W3C [Peer DID Method Specification](#). The method can be used independent of any central source of truth, and



is intended to be cheap, fast, scalable, and secure. It is suitable for most private relationships between people, organisations, and things.

- “**verificationMethod**”, which is a standard W3C VC object that specifies the Public Key associated with the DID of the employee. The binding between the DID of the employee and the Public Key was performed at the moment of credential issuance by Packet delivery.
- “**roles**” is an array with one or more role specifications. Each specification defines a potential target entity that will receive the credential, and one or more names of roles defined by that target entity.
 - “**target**” is the DID of the entity that will receive the credential.
 - “**names**” is an array with one or more roles that the target entity recognizes and that will be used by the target entity to apply its own access control policies. In the example, we have used both “*buyer*” and “*seller*” roles as defined by the Marketplace. Other entities can define their own roles for their specific purposes. Names are made unique in the ecosystem thanks to the *target* property.
- The rest of the fields in the credential have the usual meaning in the standard W3C Verifiable Credential Data Model.

The “id” field at the top level is the identification of the credential, which can be used for revocation if that functionality is required. The basic requirements for the “id” field are that:

- It is unique in the scope where it is going to be used
- It is based on a cryptographically secure random number generator and so is difficult to “guess” by a potential attacker who could try to revoke a given credential. Using such IDs reduces the probability of an attacker guessing the id to the same level than an attacker guessing a private key,
- It is not related in any way with the personal data included in the credential, to minimise the risk of correlation

A UUID Version 4 complies with all those requirements but other schemas can be used.

3.4.3.2.2 Employee of Happy Pets (or No Cheaper)

The employee credential issued by Happy Pets and No Cheaper companies to its employees are virtually identical to the employee credential from Packet Delivery described above. The main difference is the set of roles assigned to the employee and specified in the “roles” claim.

```
// Credential issued by HappyPets to its employees, providing access
// to order creation in PacketDelivery.
{
  "@context": [
    "https://www.w3.org/ns/vc#",
    "https://w3id.org/dome/claims#"
  ],
  "id": "did:example:12345678901234567890123456789012",
  "type": "VerifiableCredential",
  "issuer": "did:example:HappyPetCompany",
  "issuanceDate": "2023-01-01T12:00:00Z",
  "subject": {
    "name": "John Doe",
    "role": "Employee"
  },
  "claim": [
    {
      "type": "VerifiableCredentialClaim",
      "name": "roles",
      "value": [
        {
          "target": "did:example:CustomerOrderSystem",
          "names": [
            "buyer"
          ]
        }
      ]
    }
  ]
}
```



```

"https://www.w3.org/ns/credentials/v2",
"https://happypets.fiware.io/2022/credentials/employee/v1"
],
"id": "https://happypets.fiware.io/credentials/25159389-8dd17b796ac0",
"type": ["VerifiableCredential", "EmployeeCredential"],
"issuer": {
  "id": "did:elsi:EU.EORI.NLHAPPYPETS"
},
"issuanceDate": "2022-03-22T14:00:00Z",
"validFrom": "2022-03-22T14:00:00Z",
"expirationDate": "2023-03-22T14:00:00Z",
"credentialSubject": {
  "id": "did:peer:99ab5bca41bb45b78d242a46f0157b7d",
  "verificationMethod": [
    {
      "id": "did:peer:99ab5bca41bb45b78d242a46f0157b7d#key1",
      "type": "JwsVerificationKey2020",
      "controller": "did:peer:99ab5bca41bb45b78d242a46f0157b7d",
      "publicKeyJwk": {
        "kid": "key1",
        "kty": "EC",
        "crv": "P-256",
        "x": "IJtvoA5_XptBvcfcrtvGCvXd9bLymmfBSSdNJf5mogo",
        "y": "fSc4gZX2R3QKKfHvS3m2vGSVSN8Xc04qsquyfEM55Z0"
      }
    }
  ],
  "roles": [
    {
      "target": "did:elsi:EU.EORI.NLPACKETDEL",
      "names": ["P.Create"]
    }
  ],
  "name": "Jane Doe",
  "given_name": "Jane",
  "family_name": "Doe",
  "preferred_username": "j.doe",
  "email": "janedoe@packetdelivery.com"
}
}

```



3.4.3.2.3 Customer of Happy Pets (or No Cheaper)

This credential is used by Happy Pets to delegate access control to customers that want access to services provided by Packet Delivery and that were purchased by Happy Pets in the past.

It follows the same model as with employee credentials except that:

- The credential should be issued by Happy Pets to customers using a secure and authenticated channel created as part of a previous customer onboarding process (KYC).
- The role included in the credential corresponds to the type of customer, with the role name defined and understood by the service provider, in this case Packet Delivery.

```
// Credential issued by HappyPets to a customer,
// providing access to Gold services at PacketDelivery.
{
  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://happypets.fiware.io/2022/credentials/employee/v1"
  ],
  "id": "https://happypets.fiware.io/credentials/25159389-8dd17b796ac0",
  "type": ["VerifiableCredential", "CustomerCredential"],
  "issuer": {
    "id": "did:elsi:EU.EORI.NLHAPPYPETS"
  },
  "issuanceDate": "2022-03-22T14:00:00Z",
  "validFrom": "2022-03-22T14:00:00Z",
  "expirationDate": "2023-03-22T14:00:00Z",
  "credentialSubject": {
    "id": "did:peer:99ab5bca41bb45b78d242a46f0157b7d",
    "verificationMethod": [
      {
        "id": "did:peer:99ab5bca41bb45b78d242a46f0157b7d#key1",
        "type": "JwsVerificationKey2020",
        "controller": "did:peer:99ab5bca41bb45b78d242a46f0157b7d",
        "publicKeyJwk": {
          "kid": "key1",
          "kty": "EC",
          "crv": "P-256",
          "x": "lJtvoA5_XptBvcfcrvtGCvXd9bLymmfBSSdNJf5mogo",
          "y": "fSc4gZX2R3QKKfHvS3m2vGSVSN8Xc04qsquyfEM55Z0"
        }
      }
    ]
  }
}
```



```

    ],
    "roles": [
        {
            "target": "did:elsi:EU.EORI.NLPACKETDEL",
            "names": ["P.Info.gold"] // Or P.Info.standard
        }
    ],
    "name": "Jane Doe",
    "given_name": "Jane",
    "family_name": "Doe",
    "preferred_username": "j.doe",
    "email": "janedoe@packetdelivery.com"
}
}

```

3.4.3.2.4 Role-based access

As can be seen in the above credentials, they contain claims specifying roles. The roles are not defined by the issuer of the credential, but by the provider (i.e.: the relying party) that is going to receive the credential and perform authentication and authorization.

The provider defines a role having a certain name, and this role is mapped to a certain policy set representing the policies that the provider wants to enforce. An offering on the marketplace then just represents a certain role (or several roles). When acquiring access to an offering on the marketplace, these roles then get issued to the acquiring organisation within the Authorisation Registry of the provider. Furthermore, the acquiring organisation then can just assign these roles to their users by embedding the roles inside the Verifiable Credential issued to its users. When accessing the service, it is up to the PEP proxy/PDP component of the provider to obtain the set of attribute-based policies that belong to the assigned roles and to perform the evaluation of granting access based on the NGSI-LD request.

It is out of scope for this document to describe the actual policy language and engine used to perform the enforcement (ODRL, Rego, etc).

3.4.3.3 Detailed workflows

3.4.3.3.1 Create Offering

We now describe the process of creating an offer. In the reference use case, Packet Delivery needs to perform it twice for creating the offerings for “Basic Delivery” and “Premium Delivery”, providing a different set of offering information. The process will be performed by an employee of the Packet Delivery company.

When using the SIOP flows with Verifiable Credentials it can be observed that DOME does not have to query any other entity in the ecosystem to verify the credential because all the information needed is in the Verifiable Credential presented by the employee and in the



Decentralised Verifiable Registry (implemented in our case using a blockchain network), accessed via the Universal Resolver.

In other words, the flows are essentially peer-to-peer and do not require any centralised IdP to be queried, providing an efficient, scalable, private and resilient framework.

The following gives a detailed description of the offer creation process. Figure 6.3.a presents the different interactions in an architectural overview, whereas Figure 6.3.b shows a detailed sequence diagram of the whole process.

In the following, a description is given for each of the sequence steps.

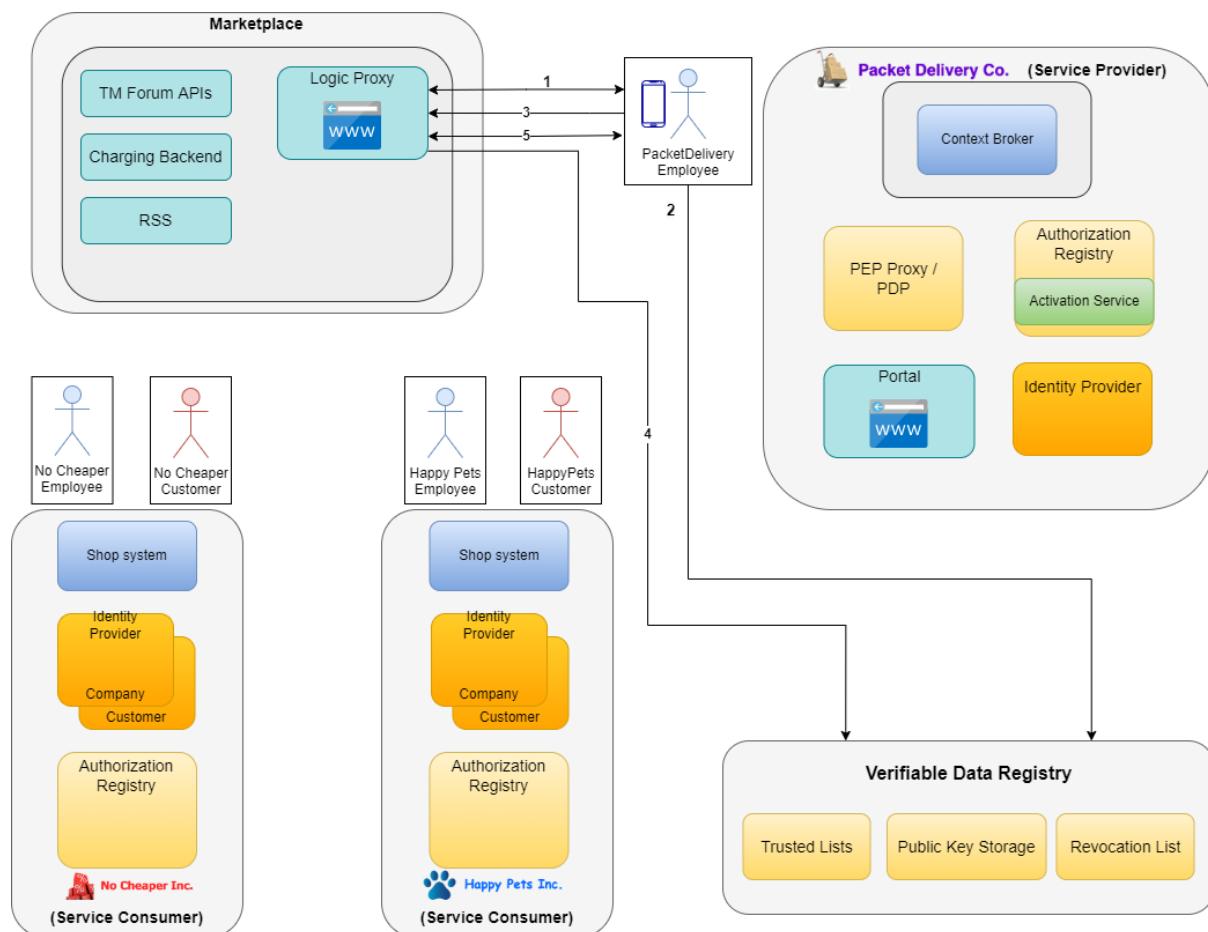


Figure 6.3.a: Architecture diagram for step “Create Offering”

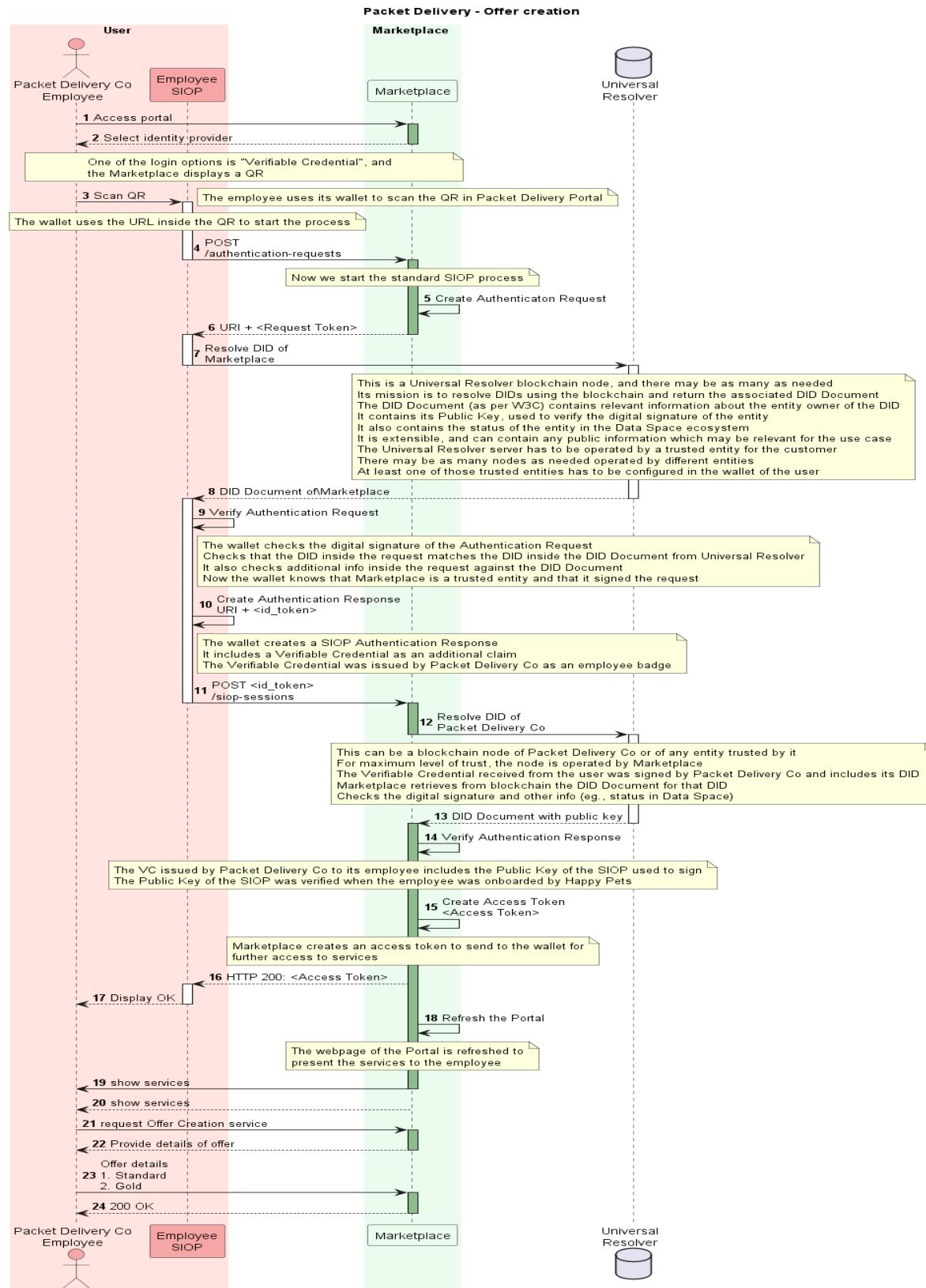


Figure 36 - Sequence diagram for step “Create Offering”



1. A Packet Delivery employee accesses the Marketplace portal (DOME), in order to login.
2. The Marketplace portal displays a list of Identity Providers for selecting the desired Identity Provider for login. One of the login options is “Login with Verifiable Credentials”.
3. Packet Delivery Co employee selects the “Verifiable Credentials” login method, which causes the Marketplace portal to generate a QR containing the URL of the /authentication-requests endpoint of the Marketplace server.
4. The employee scans the QR with her mobile and the mobile calls the /authentication-requests endpoint.
5. This starts a standard SIOP (Self-Issued OpenID Provider) flow, where the Marketplace plays the role of Relying Party (RP in Open ID Connect terminology) and the mobile device of the employee as a Self-Issued IDP. In this step, Marketplace creates a SIOP Authentication Request. As a Self-Issued OP may be running as a native application or progressive web application (PWA), the RP may not have a network-addressable endpoint to communicate directly with the OP. We have to leverage the implicit flow of OpenID Connect to communicate with such locally-running OPs, as described in https://openid.net/specs/openid-connect-self-issued-v2-1_0.html.

The Authentication Request travels in the response to the HTTP GET request performed in the previous point, as a JWT signed by Marketplace. The decoded contents of the JWT may be:

```
openid://?
scope=openid
&response_type=id_token
&response_mode=post
&client_id=did:elsi:EU.EORI.NLMARKETPLA
&redirect_uri=https://marketplace.dome.org/siop_sessions
&claims=... //the Marketplace would specify here what type of claims it wants the employee to provide. Those claims should be connected to roles of users in the application, documented in the marketplace
&registration={
    "subject_syntax_types_supported": ["did:key",
        "urn:ietf:params:oauth:jwk-thumbprint"]
}
&nonce=n-0S6_WzA2Mj
```



6. The Authentication Request is returned to the employee wallet acting as SIOP. The SIOP flow uses a new response mode **post** which is used to request the SIOP to deliver the result of the authentication process to a certain endpoint. The parameter **response_mode** is used to carry this value.

This endpoint where the SIOP shall deliver the authentication result is defined in the standard parameter **redirect_uri**.

7. In this step the employee verifies that the Marketplace is a trusted entity belonging to the ecosystem, by resolving the DID of the Marketplace which is received in the **client_id** parameter of the Authentication Request.

To resolve a DID, the wallet sends a GET request to the **/api/did/v1/identifiers/did:elsi:EU.EORI.NLMARKETPLA** endpoint of one of several trusted servers implementing the Universal Resolver functionality. The Universal Resolver includes a blockchain node, and there may be as many as needed. Its mission is to resolve DIDs using the blockchain and return the associated DID Document. The DID Document (as per W3C) contains relevant information about the entity owner of the DID. It contains its Public Key, used to verify the digital signature of the entity. It also contains the status of the entity in the Data Space ecosystem. It is extensible and can contain any public information which may be relevant for the use case. The Universal Resolver server must be operated by a trusted entity for the customer. There may be as many nodes as needed operated by different entities. At least one of those trusted entities has to be configured in the wallet of the employee.

8. The wallet receives the DID Document of Marketplace, with trusted information about the entity, including the Public Key associated with the Private Key that Marketplace uses to digitally sign tokens. For example:

```
{
  "payload": {
    "@context": [
      "https://www.w3.org/ns/did/v1",
      "https://w3id.org/security/v1"
    ],
    "id": "did:elsi:EU.EORI.NLMARKETPLA",
    "verificationMethod": [
      {
        "id": "did:elsi:EU.EORI.NLMARKETPLA#key-verification",
        "type": "JwsVerificationKey2020",
        "controller": "did:elsi:EU.EORI.NLMARKETPLA",
        "publicKeyJwk": {
          "kid": "key-verification",
          "kty": "EC",
          "crv": "secp256k1",
          "x": "V8XptJkb5wpIYkExcTF4nkyYVp7t5H5d5C4UPqCCM9c",
          "y": "kn3nSPxIlvd9iaG0N4v14ceuo8E4PcLXhhGeDzCE7VM"
        }
      }
    ]
  }
}
```



```

        }
    },
],
"service": [
{
    "id": "did:elsi:EU.EORI.NLMARKETPLA#info",
    "type": "EntityCommercialInfo",
    "serviceEndpoint": "https://marketplace.fiware.io/info",
    "name": "Packet Delivery co."
},
{
    "id": "did:elsi:EU.EORI.NLMARKETPLA#sms",
    "type": "SecureMessagingService",
    "serviceEndpoint": "https://marketplace.fiware.io/api/sms"
}
],
"anchors": [
{
    "id": "redt.alastria",
    "resolution": "UniversalResolver",
    "domain": "marketplace.dataspace",
    "ethereumAddress": "0xbcb9b29eeb28f36fd84f1CfF98C3F1887D831d78"
}
],
"created": "2021-11-14T13:02:37Z",
"updated": "2021-11-14T13:02:37Z"
}
}
}

```

9. The DID Document includes one or more public keys inside the “verificationMethod” array. The keys are identified by the “id” field in each element of the array. The employee wallet uses the **kid** field that was received in the Authentication Request (in the protected header of the JWT) to select the corresponding Public Key and verify the signature of the JWT. It also verifies that the top-level “**id**” field in the DID Document (“did:elsi:EU.EORI.NLMARKETPLA”) is equal to the **client_id** parameter of the Authentication Request.
10. The employee wallet creates an Authentication Response to be posted in the **redirect_uri** specified by Marketplace in step 5. The contents of the Authentication Response are described below.
11. The SIOP sends the authentication response to the endpoint passed in the **redirect_uri** authentication request parameter using a HTTP POST request using “application/x-www-form-urlencoded” encoding. The response contains an ID Token and a VP (Verifiable Presentation) token as defined in [OpenID for Verifiable Presentations](#).



```
POST /siop_sessions HTTP/1.1
Host: marketplace.dome.org
Content-Type: application/x-www-form-urlencoded

id_token=eyJ0...NiJ9eyJ1c...I6ljifX0.DeWt4Qu...ZXso
&vp_token=...
&state=af0ifjsldkj
```

The decoded **id_token** would be:

```
{
  "iss": "https://self-issued.me/v2",
  "aud": "did:elsi:EU.EORI.NLMARKETPLA",
  "iat": 1615910538,
  "exp": 1615911138,
  "sub": "did:peer:99ab5bca41bb45b78d242a46f0157b7d",
  "auth_time": 1615910535,
  "nonce": "n-0S6_WzA2Mj"
}
```

The **sub** claim is *did:peer:99ab5bca41bb45b78d242a46f0157b7d* which is the DID of the user and for privacy reasons it is not registered in any blockchain or centralized repository. It must be the same as the DID included in the Verifiable Credential that was issued by the Packet Delivery company when onboarding the employee and which travels in the authentication response.

The **vp_token** includes the Verifiable Presentation, which can be in two formats: **jwt_vp** (JWT encoded) or **ldp_vp** (JSON-LD encoded). The following example is using the JWT encoding:

```
{
  "format": "jwt_vp",
  "presentation": "eyJhbGciOiJSUzI1NilsInR5cCl6IkpxVCIsImtpZCI6ImRpZDpleGFtcGxIOMFiZmUxM2Y3MTIxMjA0
  MzFjMjc2ZTEyZWNhYiNrZXlzMTEifQ.eyJzdWliOiJkaWQ6ZXhhbXBsZTpIYmZlYjFmNzEyZWJjNmYxY
  zl3NmUxMmVjMjEiLCJqdGkiOiJodHRwOi8vZXhhbXBsZS5lZHUVY3JIZGVudGlhbHMvMzczzMilsImlz
  yI6Imh0dHBzOi8vZXhhbXBsZS5jb20va2V5cy9mb28uandriwibmJmljoxNTQxNDkzNzI0LCJpYXQiO
  jE1NDE0OTM3MjQsImV4cCl6MTU3MzAyOTcMywibm9uY2UiOil2NjAhNjM0NUTZXiLCJ2Yyl6eyJAY
  29udGV4dCl6WyJodHRwczovL3d3dy53My5vcmcvMjAxOC9jcmVkJW50aWFscy92MSIsImh0dHBzOi8vd
  3d3LnczLm9yZy8yMDE4L2NyZWRlbnRpYWxzL2V4YW1wbGVzL3YxI0slnR5cGUIOlsivmVyaWZpYWJsZ
  UnYzWRlbnRpYWwiLCJVbml2ZXJzaXR5RGVncmVIQ3JIZGVudGlhbCJdLCJcmVkJW50aWFsU3ViamVjd
  CI6eyJkZWdyZWUiOnsidHlwZSI6IkJhY2hlbG9yRGVncmVliiwibmFtZSI6IjxzcGFulGxhbmcc9J2ZyL
```



```

UNBJz5CYWNjYWxhdXLDqWF0IGVulG11c2IxdWVzIG51bcOpcmlxdWVzPC9zcGFuPiJ9fX19.KLJo5GAy
BND3LDTn9H7FQoKEsUEi8jKwXhGvoN3JtRa51xrNDgXD0cq1UTYB-rK4Ft9YVmR1NI_ZOF8oGc_7wAp
8PHbF2HaWodQloOBxxT-4WNqAxft7ET6lkH-4S6Ux3rSGAmczMohEEf8eCeN-jC8WekdPl6zKZQj0YPB
1rx6X0-xIFBs7cl6Wt8rfBP_tZ9YgVWrQmUWypSioc0MUyiphmyEbLZagTyPIUyflGIEdqrZAv6eSe6R
txJy6M1-ID7a5HTzanYTWBPAUHDZGyGKXdJw-W_x0IWChBzl8t3kpG253fg6V3tPgHeKXE94fz_QpYfg
--7kLsyBAfQGbg"
}

```

Which decoded would be:

```
{
  "@context": ["https://www.w3.org/2018/credentials/v1"],
  "type": ["VerifiablePresentation"],
  "verifiableCredential": [
    {
      "@context": [
        "https://www.w3.org/ns/credentials/v2",
        "https://marketplace.dome.org/2022/credentials/employee/v1"
      ],
      "id": "https://pdc.fiware.io/credentials/6e14b8b8-87fa0014fe2a",
      "type": ["VerifiableCredential", "EmployeeCredential"],
      "issuer": {
        "id": "did:elsi:EU.EORI.NLPACKETDEL"
      },
      "issuanceDate": "2022-03-22T14:00:00Z",
      "validFrom": "2022-03-22T14:00:00Z",
      "expirationDate": "2023-03-22T14:00:00Z",
      "credentialSubject": {
        "id": "did:peer:99ab5bca41bb45b78d242a46f0157b7d",
        "verificationMethod": [
          {
            "id": "did:peer:99ab5bca41bb45b78d242a46f0157b7d#key1",
            "type": "JwsVerificationKey2020",
            "controller": "did:peer:99ab5bca41bb45b78d242a46f0157b7d",
            "publicKeyJwk": {
              "kid": "key1",
              "kty": "EC",
              "crv": "P-256",
              "x": "IJtvoA5_XptBvcfcrvtGCvXd9bLymmfBSSdNJf5mogo",
              "y": "fSc4gZX2R3QKKfHvS3m2vGSVSN8Xc04qsquyfEM55Z0"
            }
          }
        ],
        "roles": [

```



```
{
  "target": "did:elsi:EU.EORI.NLMARKETPLA",
  "names": ["seller", "buyer"]
}
],
"name": "Jane Doe",
"given_name": "Jane",
"family_name": "Doe",
"preferred_username": "j.doe",
"email": "janedoe@packetdelivery.com"
}
}
]
```

12. Marketplace uses its own blockchain node or the one from a trusted entity implementing the Universal Resolver functionality to resolve the DID of Packet Delivery Co, which is inside the Verifiable Credential received in the Verifiable Presentation. This DID can be found in the “issuer” field of the “verifiableCredential” structure above. Resolution is performed sending a GET request to the Universal Resolver: **/api/did/v1/identifiers/did:elsi:EU.EORI.NLPACKETDEL**
Marketplace could use a Universal Resolver operated by a different entity, but this would reduce the level of trust compared to using its own server directly connected to the blockchain network.
13. Marketplace receives the DID Document of Packet Delivery Co with trusted information about the company, including the Public Key associated with the Private Key that Packet Delivery Co used to digitally sign the Verifiable Credential that the employee has just sent inside a Verifiable Presentation as part of the authentication flow. **Using the Public Key and the DID inside the DID Document, it can verify the signature of the Verifiable Credential and that Packet Delivery Co is a trusted entity in the ecosystem and that it is active.**
14. The above is just for verification of the Verifiable Credential. In addition, Marketplace can also verify that the Verifiable Presentation including the Verifiable Credential is sent by the employee and not by a malicious agent. To do so, it uses the Public Key of the employee in the “verificationMethod” of the “credentialSubject” structure. That public key is cryptographically bound to the employee DID during the onboarding process that Packet Delivery Co performed with its employee.
15. Once all verifications have been performed, Marketplace creates an Access Token for the employee so she can use it to access services in the Marketplace server in the future.



16. The wallet (SIOP) receives the access token and saves it temporarily to be able to request services from Marketplace.
17. The wallet displays a success message to the employee.
18. The Marketplace server refreshes the page (it was the login page before) and displays the services available to the employee of Packet Delivery Co.

At this point the Packet Delivery Co employee is logged in on the Marketplace application. The user is now able to create catalogues, products and offerings.

At this moment, the Marketplace knows the following:

- That Packet Delivery Co belongs to the Data Space and can issue credentials of the type EmployeeCredential because it is included in the Trusted Issuers Registry and is active, because this info is in the DID Document retrieved in step 13.
- That Packet Delivery Co says that the user is one of its employees. This info is inside the Verifiable Credential that is digitally signed by Packet Delivery Co.

From this point on, the Marketplace can display to the user the services available to her and execute them if the user is entitled to do so. The Marketplace can use all the claims inside the credential to perform RBAC/ABAC access control and policy enforcement.

3.4.3.3.2 Acquisition of rights / Activation

The process of acquiring access to the packet delivery service is displayed. It is performed by employees of both parties separately, Happy Pets and No Cheaper, where the former one acquires access to the “Premium Delivery” offering and the latter acquires the “Basic Delivery” offering.

In the following we describe the authentication process for employees of Happy Pets and No Cheaper, which is similar to the one described above for the employee of Packet Delivery.



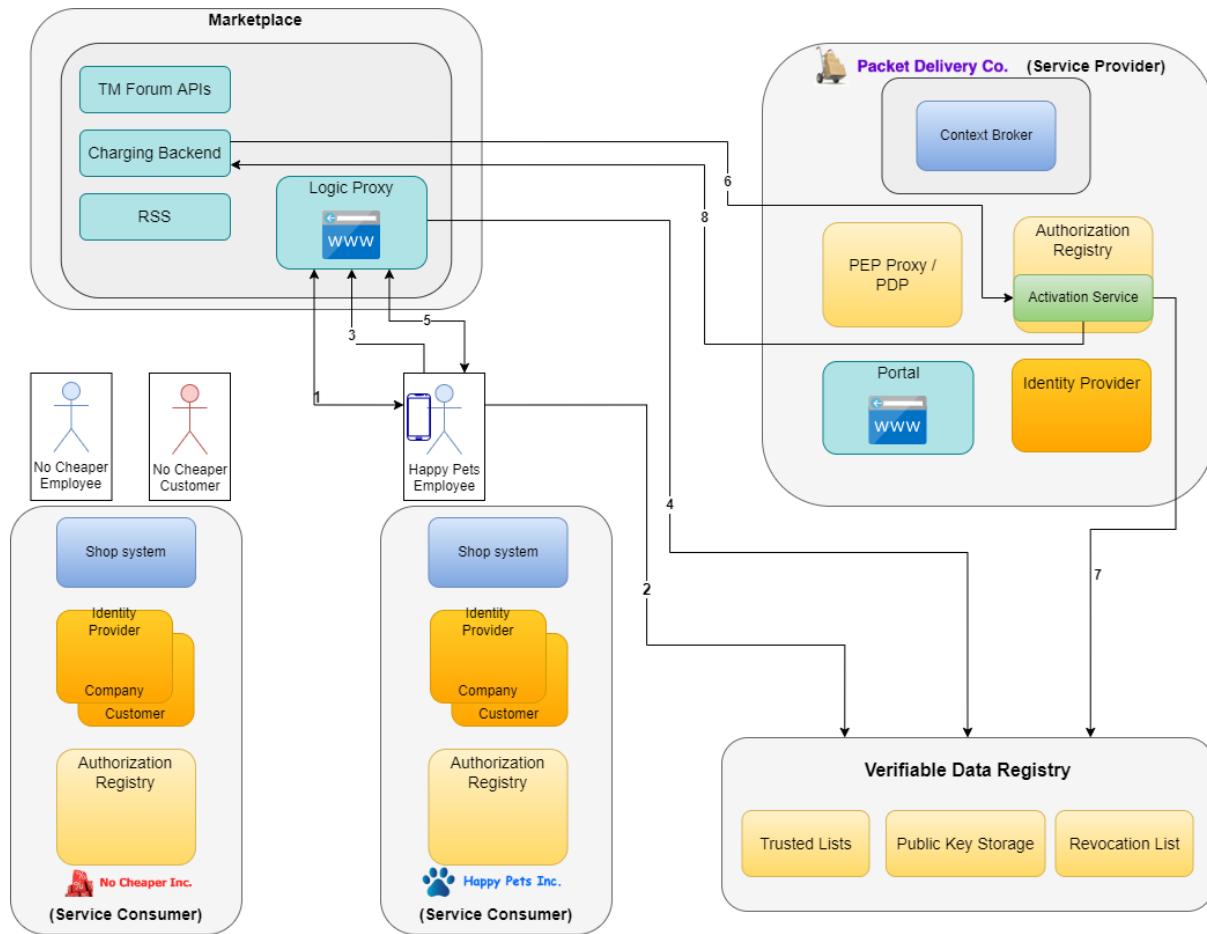
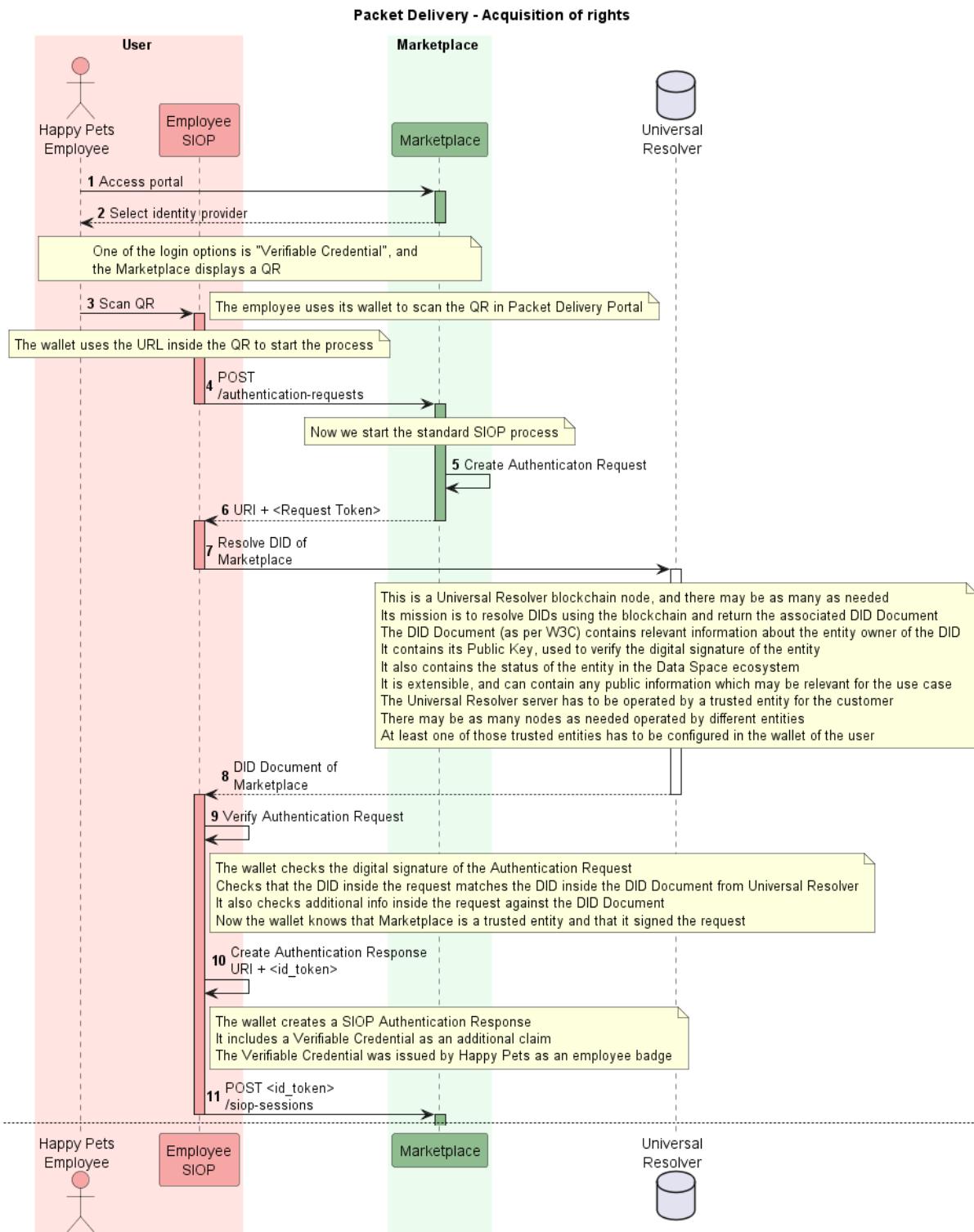


Figure 6.4: Sequence diagram for step “Acquisition of Rights / Activation”



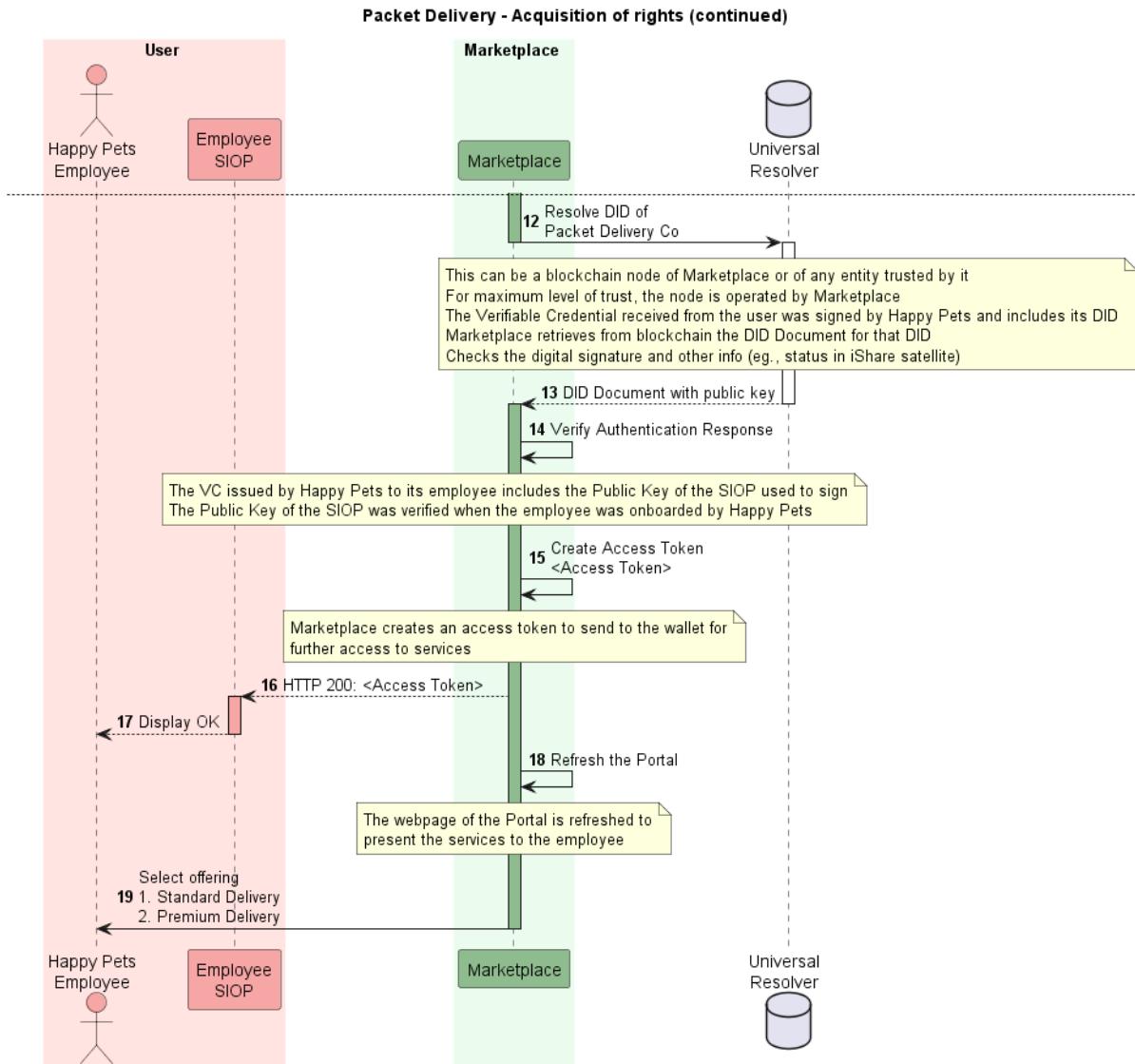


Figure 37 - Sequence diagram for step “Acquisition of Rights / Activation”

19. The Happy Pets employee accesses the Marketplace portal (DOME), in order to login.
20. Happy Pets employee is displayed a list of Identity Providers for selecting the desired Identity Provider for login. Happy Pets employee gets forwarded to a page for selecting the desired Identity Provider for login. One of the login options is “Verifiable Credentials” or something similar.
21. Happy Pets employee selects the “Verifiable Credentials” login method, which causes the Marketplace portal to generate a QR containing the URL of the /authentication-requests endpoint of the Marketplace server.
22. The employee scans the QR with her mobile and the mobile calls the /authentication-requests endpoint.

23. This starts a standard SIOP (Self-Issued OpenID Provider) flow, where the Marketplace IDP plays the role of Relying Party (RP in Open ID Connect terminology) and the mobile device of the employee as a Self-Issued IDP. In this step, Marketplace IDP creates a SIOP Authentication Request. As a Self-Issued OP may be running locally as a native application or progressive web application (PWA), the RP may not have a network-addressable endpoint to communicate directly with the OP. We have to leverage the implicit flow of OpenID Connect to communicate with such locally-running OPs, as described in https://openid.net/specs/openid-connect-self-issued-v2-1_0.html.

The Authentication Request travels in the response to the HTTP GET request performed in the previous point, as a JWT signed by Packet Delivery company. The decoded contents of the JWT may be:

```
openid://?
scope=openid
&response_type=id_token
&response_mode=post
&client_id=did:elsi:EU.EORI.NLMARKETPLA
&redirect_uri=https://marketplace.dome.org/siop_sessions
&claims=...
&registration={
  "subject_syntax_types_supported": ["did:key",
  "urn:ietf:params:oauth:jwk-thumbprint"]
}
&nonce=n-0S6_WzA2Mj
```

24. The Authentication Request is returned to the employee wallet acting as SIOP. The SIOP flow uses a new response mode **post** which is used to request the SIOP to deliver the result of the authentication process to a certain endpoint. The parameter **response_mode** is used to carry this value.

This endpoint where the SIOP shall deliver the authentication result is defined in the standard parameter **redirect_uri**.

25. In this step the employee verifies that the Marketplace is a trusted entity belonging to the ecosystem, by resolving the DID of the Marketplace which is received in the **client_id** parameter of the Authentication Request.

To resolve a DID, the wallet sends a GET request to the **/api/did/v1/identifiers/did:elsi:EU.EORI.NLMARKETPLA** endpoint of one of several trusted servers implementing the Universal Resolver functionality. The Universal Resolver includes a blockchain node, and there may be as many as needed. Its mission is to resolve DIDs using the blockchain and return the associated DID Document. The DID Document (as per W3C) contains relevant information about the entity owner of the DID. It contains its Public Key, used to verify the digital



signature of the entity. It also contains the status of the entity in the Data Space ecosystem. It is extensible and can contain any public information which may be relevant for the use case. The Universal Resolver server must be operated by a trusted entity for the customer. There may be as many nodes as needed operated by different entities. At least one of those trusted entities has to be configured in the wallet of the employee.

26. The wallet receives the DID Document of Marketplace, with trusted information about the entity, including the Public Key associated with the Private Key that Marketplace uses to digitally sign tokens. For example:

```
{
  "payload": {
    "@context": [
      "https://www.w3.org/ns/did/v1",
      "https://w3id.org/security/v1"
    ],
    "id": "did:elsi:EU.EORI.NLMARKETPLA",
    "verificationMethod": [
      {
        "id": "did:elsi:EU.EORI.NLMARKETPLA#key-verification",
        "type": "JwsVerificationKey2020",
        "controller": "did:elsi:EU.EORI.NLMARKETPLA",
        "publicKeyJwk": {
          "kid": "key-verification",
          "kty": "EC",
          "crv": "secp256k1",
          "x": "V8XptJkb5wpIYkExcTF4nkyYVp7t5H5d5C4UPqCCM9c",
          "y": "kn3nSPxIlvd9iaG0N4v14ceuo8E4PcLXhhGeDzCE7VM"
        }
      }
    ],
    "service": [
      {
        "id": "did:elsi:EU.EORI.NLMARKETPLA#info",
        "type": "EntityCommercialInfo",
        "serviceEndpoint": "https://marketplace.dome.org/info",
        "name": "Packet Delivery co."
      },
      {
        "id": "did:elsi:EU.EORI.NLMARKETPLA#sms",
        "type": "SecureMessagingService",
        "serviceEndpoint": "https://marketplace.fiware.io/api/sms"
      }
    ],
    "anchors": [
      {
        "id": "redt.alastria",
        "resolution": "UniversalResolver",
        "domain": "marketplace.dataspace",
        "ethereumAddress": "0xbcb9b29eeb28f36fd84f1CfF98C3F1887D831d78"
      }
    ],
  }
}
```



```

    "created": "2021-11-14T13:02:37Z",
    "updated": "2021-11-14T13:02:37Z"
}
}

```

27. The DID Document includes one or more public keys inside the “verificationMethod” array. The keys are identified by the “id” field in each element of the array. The employee wallet uses the **kid** field that was received in the Authentication Request (in the protected header of the JWT) to select the corresponding Public Key and verify the signature of the JWT. It also verifies that the top-level “**id**” field in the DID Document (“did:elsi:EU.EORI.NLMARKETPLA”) is equal to the **client_id** parameter of the Authentication Request.
28. The employee wallet creates an Authentication Response to be posted in the **redirect_uri** specified by Marketplace in step 5. The contents of the Authentication Response are described below.
29. The SIOP sends the authentication response to the endpoint passed in the **redirect_uri** authentication request parameter using a HTTP POST request using “application/x-www-form-urlencoded” encoding. The response contains an ID Token and a VP (Verifiable Presentation) token as defined in https://openid.net/specs/openid-connect-4-verifiable-presentations-1_0.html.

```

POST /siop_sessions HTTP/1.1
Host: marketplace.dome.org
Content-Type: application/x-www-form-urlencoded

id_token=eyJ0 ... NiJ9eyJ1c ... I6ljlifX0.DeWt4Qu ... ZXso
&vp_token=...
&state=af0ifjsldkj

```

The decoded **id_token** would be:

```

{
  "iss": "https://self-issued.me/v2",
  "aud": "did:elsi:EU.EORI.NLMARKETPLA",
  "iat": 1615910538,
  "exp": 1615911138,
  "sub": "did:peer:99ab5bca41bb45b78d242a46f0157b7d",
  "auth_time": 1615910535,
  "nonce": "n-0S6_WzA2Mj",
}

```

The **sub** claim is *did:peer:99ab5bca41bb45b78d242a46f0157b7d* which is the DID of the user and for privacy reasons it is **not registered in any blockchain or**



centralised repository. It must be the same as the DID included in the Verifiable Credential that was issued by the Happy Pets company when onboarding the employee and which travels in the authentication response.

The **vp_token** includes the Verifiable Presentation, which can be in two formats: **jwt_vp** (JWT encoded) or **ldp_vp** (JSON-LD encoded). The following example is using the JWT encoding:

```
{  
  "format": "jwt_vp",  
  "presentation":  
    "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVClsImtpZCI6ImRpZDpleGFtcGxIOMfzUmUxM2Y3MTIxMjA0  
MzFjMjc2ZTEyZWNhYiNrZXlzLTEifQ.eyJzdWlIoiJkaWQ6ZXhhbXBsZTpIYmZlYjFmNzEyZWJNmYxY  
zl3NmUxMmVjMjEiLCJqdGkiOiJodHRwOi8vZXhhbXBsZS5lZHUVY3JIZGVudGlhbHMvMzczMilsImlc  
_yI6Imh0dHBzOi8vZXhhbXBsZS5jb20va2V5cy9mb28uandrliwibmJmljoxNTQxNDkzNzI0LCJpYXQiO  
jE1NDE0OTM3MjQsImV4cCI6MTU3MzAyOTcyMywibm9uY2UiOi2NjAhNjM0NUTZXliLCJ2YyI6eyJA  
Y29udGV4dCI6WyJodHRwczovL3d3dy53My5vcmcvMjAxOC9jcmVkJZW50aWFscy92MSIsImh0dHBzO  
i8vd3d3LnczLm9yYz8yMDE4L2NyZWRlbnRpYWxzL2V4YW1wbGVzL3YxII0sInR5cGUiOlsiVmVyaWZ  
pYWJsZUNyZWRlbnRpYWwiLCJVbml2ZXJzaXR5RGVncmVIQ3JIZGVudGlhbCJdLCJcmVkJZW50aW  
FsU3ViamVjdCl6eyJkZWdyZWUiOnsidHlwZSI6IkjhY2hlB9yRGVncmVliwibmFtZSI6ljxzcGFulGxhb  
mc9j2ZyLUNBjz5CYWNjYWxhdXLDqWF0IGVuIG11c2IxdWVzIG51bcOpclmxWVzPC9zcGFuPiJ9fX  
19.KLJo5GAy BND3LDTn9H7FQokEsUEi8jKwXhGvoN3JtRa51xrNDgXD0cq1UTYB-  
rK4Ft9YVmR1NI_ZOF8oGc_7wAp8PHbF2HaWodQloOBxxT-4WNqAxft7ET6IkH-  
4S6Ux3rSGAmczMohEEf8eCeN-jC8WekdPl6zKZQj0YPB1rx6X0-  
xIFBs7cl6Wt8rfBP_tZ9YgVWrQmUWypSioc0MUyiphmyEbLZagTyPIUyfGIEdqrZAv6eSe6R  
txJy6M1-ID7a5HTzanYTWBPAUHDZGyGKXdJw-  
W_x0IWChBzl8t3kpG253fg6V3tPgHeKXE94fz_QpYfg  
--7kLsyBAfQGbg"  
}
```

Which decoded could be:

```
{
  "@context": ["https://www.w3.org/2018/credentials/v1"],
  "type": ["VerifiablePresentation"],
  "verifiableCredential": [
    {
      "@context": [
        "https://www.w3.org/ns/credentials/v2",
        "https://happypets.fiware.io/2022/credentials/employee/v1"
      ],
      "id": "https://happypets.fiware.io/credentials/25159389-8dd17b796ac0",
      "type": ["VerifiableCredential", "EmployeeCredential"],
      "issuer": {
        "id": "did:elsi:EU.EORI.NLHAPPYPETS"
      },
      "issuanceDate": "2022-03-22T14:00:00Z",
      "validFrom": "2022-03-22T14:00:00Z",
      "expirationDate": "2023-03-22T14:00:00Z",
      "credentialSubject": {
```

```

    "id": "did:peer:99ab5bca41bb45b78d242a46f0157b7d",
    "verificationMethod": [
        {
            "id": "did:peer:99ab5bca41bb45b78d242a46f0157b7d#key1",
            "type": "JwsVerificationKey2020",
            "controller": "did:peer:99ab5bca41bb45b78d242a46f0157b7d",
            "publicKeyJwk": {
                "kid": "key1",
                "kty": "EC",
                "crv": "P-256",
                "x": "IjtvoA5_XptBvcfcrtGCvXd9bLymmfBSSdNJf5mogo",
                "y": "fSc4gZX2R3QKKfHvS3m2vGSVSN8Xc04qsquyfEM55Z0"
            }
        }
    ],
    "roles": [
        {
            "target": "did:elsi:EU.EORI.NLPACKETDEL",
            "names": ["P.Create"]
        }
    ],
    "name": "Jane Doe",
    "given_name": "Jane",
    "family_name": "Doe",
    "preferred_username": "j.doe",
    "email": "janedoe@packetdelivery.com"
}
]
}
]
}

```

30. Marketplace uses its own blockchain node or the one from a trusted entity implementing the Universal Resolver functionality to resolve the DID of Happy Pets, which is inside the Verifiable Credential received in the Verifiable Presentation. This DID can be found in the “issuer” field of the “verifiableCredential” structure above.

Resolution is performed sending a GET request to the Universal Resolver:
/api/did/v1/identifiers/did:elsi:EU.EORI.NLHAPPYPETS

Marketplace could use a Universal Resolver operated by a different entity, but this would reduce the level of trust compared to using its own server directly connected to the blockchain network.

31. Marketplace receives the DID Document of Happy Pets with trusted information about the company, including the Public Key associated with the Private Key that Happy Pets used to digitally sign the Verifiable Credential that the employee has just sent inside a Verifiable Presentation as part of the authentication flow. **Using the**



Public Key and the DID inside the DID Document, it can verify the signature of the Verifiable Credential and that Happy Pets is a trusted entity in the ecosystem and that it is active.

32. The above is just for verification of the Verifiable Credential. In addition, Marketplace can also verify that the Verifiable Presentation including the Verifiable Credential is sent by the employee and not by a malicious agent. To do so, it uses the Public Key of the employee in the “verificationMethod” of the “credentialSubject” structure. That public key is cryptographically bound to the employee DID during the onboarding process that Happy Pets performed with its employee.
33. Once all verifications have been performed, Marketplace creates an Access Token for the employee so she can use it to access services in the Marketplace server in the future.
34. The wallet (SIOP) receives the access token and saves it temporarily to be able to request services from Marketplace.
35. The wallet displays a success message to the employee.
36. The Marketplace server refreshes the page (it was the login page before) and displays the services available to the employee of Packet Delivery Co.

At this point the Happy Pets employee is logged in on DOME. The user is now able to use the services available to her.

At this moment, DOME knows the following:

- That Happy Pets belongs to the Data Space and can issue credentials of the type EmployeeCredential because it is included in the corresponding Trusted Issuers Registry and is active, because this info is in the DID Document retrieved in step 13. Maintenance of this information is performed by the Trust Anchor entity (or entities) responsible for the Trusted Issuers Registry.
- That Happy Pets says that the user is one of its employees. This info is inside the Verifiable Credential that is digitally signed by Happy Pets.

From this point on, DOME can display to the user the services available to her and execute them if the user is entitled to do so. DOME can use all the claims inside the credential to perform RBAC/ABAC access control and policy enforcement.

Now, when acquiring access to the premium data service offering of Packet Delivery, DOME will create the necessary access policies (or roles) at the Authorization Registry of Packet Delivery, stating that Happy Pets is allowed to issue credentials of the gold customer type. The Authorization Registry implements Policy Administration Point (PAP) and Policy Management Point (PMP) functionalities.



It is out of scope for this document to describe the actual policy language and how access to the registry is performed.

For No Cheaper, the process is exactly the same as for the acquisition process for Happy Pets, except that the entities involved are No Cheaper Ltd and its employees, and that the basic offering is acquired. We do not provide a detailed flow to avoid repetition.

3.4.3.3.3 Access to data service

The process of changing the PTA attribute of a packet delivery order via the packet delivery portal is explained. The process would be similar, when trying to change the PDA or delivery address.

In the following the sequences are shown for the scenario of the Happy Pets customer changing the PTA of the delivery order. In the case of the No Cheaper customer, the sequences would be the same with the only difference being that the request for changing the PTA would be denied.

The following gives a detailed description of the process of changing the PTA attribute by the Happy Pets customer, when using Verifiable Credentials. Figure 4.4.3.1b shows a detailed sequence diagram of the whole process. The numberings in the architectural overview map to the different steps of the sequence diagram.

In the following, a description is given for each of the sequence steps.



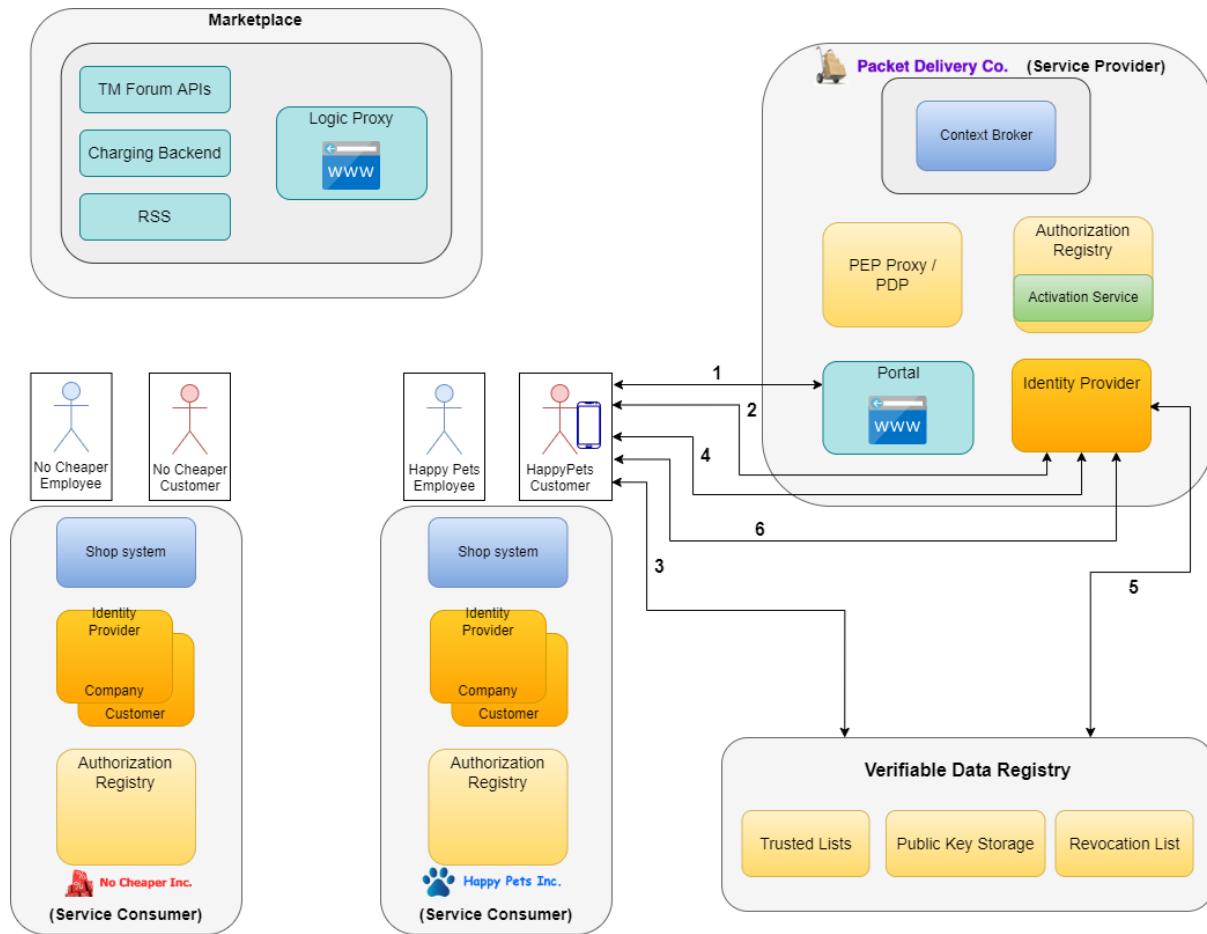


Figure 38 - Architecture diagram for step “Change PTA by Happy Pets customer”

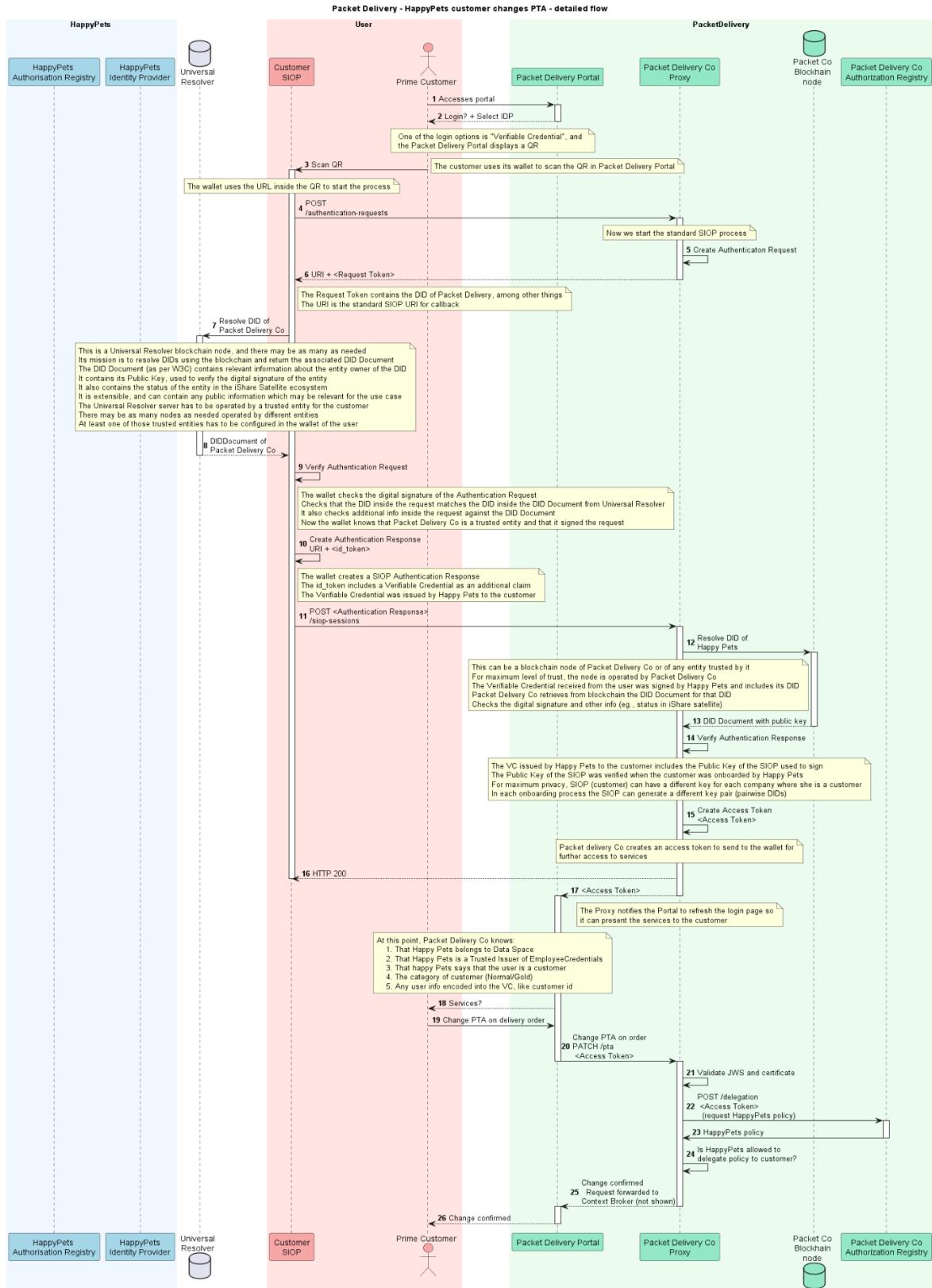


Figure 39 - Sequence diagram for step “Change PTA by Happy Pets customer”



27. Happy Pets customer accesses the Packet delivery company portal or starts the Packet Delivery company app in its smartphone, to login.
28. Happy Pets customer gets forwarded to a page for selecting the desired Identity Provider for login. One of the login options is “Verifiable Credentials” or something similar.
29. Happy Pets customer selects the “Verifiable Credentials” login method, which causes the Packet delivery company portal to generate a QR containing inside the URL of the /authentication-requests endpoint of the Packet Delivery company IDP.
30. The customer scans the QR with her mobile and the mobile calls the /authentication-requests endpoint.
31. This starts a standard SIOP (Self-Issued OpenID Provider) flow, where the Packet Delivery company IDP plays the role of Relying Party (RP in Open ID Connect terminology) and the mobile device of the customer as a Self-Issued IDP. In this step, Packet Delivery company IDP creates a SIOP Authentication Request. As a Self-Issued OP may be running locally as a native application or progressive web application (PWA), the RP may not have a network-addressable endpoint to communicate directly with the OP. We have to leverage the implicit flow of OpenID Connect to communicate with such locally-running Ops, as described in https://openid.net/specs/openid-connect-self-issued-v2-1_0.html.

The Authentication Request travels in the response to the HTTP GET request performed in the previous point, as a JWT signed by Packet Delivery company. The decoded contents of the JWT may be:

```
openid://?
  response_type=id_token
  &response_mode=post
  &client_id=did:elsi:EU.EORI.NLPACKETDEL
  &redirect_uri=https%3A%2F%2Fidp-pdc.firebaseio.io%2Fsio_sessions
  &scope=openid%20profile
  &state=af0ifjsldkj
  &nonce=n-0S6_WzA2Mj
  &registration=%7B%22subject_syntax_types_supported%22:%5B%22did%22%5D,
  %22id_token_signing_alg_values_supported%22:%5B%22RS256%22%5D%7
```

32. The Authentication Request is returned to the customer wallet acting as SIOP. The SIOP flow uses a new response mode **post** which is used to request the SIOP to deliver the result of the authentication process to a certain endpoint. The parameter **response_mode** is used to carry this value.



This endpoint where the SIOB shall deliver the authentication result is defined in the standard parameter **redirect_uri**.

33. In this step the customer verifies that the Packet Delivery company is a trusted entity belonging to the ecosystem, by resolving the DID of the Packet Delivery company which is received in the **client_id** parameter of the Authentication Request.

To resolve a DID, the wallet sends a GET request to the **/api/did/v1/identifiers/did:elsi:EU.EORI.NLPACKETDEL** endpoint of one of several trusted servers implementing the Universal Resolver functionality. The Universal Resolver includes a blockchain node, and there may be as many as needed. Its mission is to resolve DIDs using the blockchain and return the associated DID Document. The DID Document (as per W3C) contains relevant information about the entity owner of the DID. It contains its Public Key, used to verify the digital signature of the entity. It also contains the status of the entity in the Data Space ecosystem. It is extensible and can contain any public information which may be relevant for the use case. The Universal Resolver server must be operated by a trusted entity for the customer. There may be as many nodes as needed operated by different entities. At least one of those trusted entities has to be configured in the wallet of the user.

34. The wallet receives the DID Document of Packet Delivery company, with trusted information about the company, including the Public Key associated with the Private Key that Packet Delivery company uses to digitally sign tokens. For example:

```
{
  "payload": {
    "@context": [
      "https://www.w3.org/ns/did/v1",
      "https://w3id.org/security/v1"
    ],
    "id": "did:elsi:EU.EORI.NLPACKETDEL",
    "verificationMethod": [
      {
        "id": "did:elsi:EU.EORI.NLPACKETDEL#key-verification",
        "type": "JwsVerificationKey2020",
        "controller": "did:elsi:EU.EORI.NLPACKETDEL",
        "publicKeyJwk": {
          "kid": "key-verification",
          "kty": "EC",
          "crv": "secp256k1",
          "x": "V8XptJkb5wpIYkExcTF4nkyYVp7t5H5d5C4UPqCCM9c",
          "y": "kn3nSPxIlvd9iaG0N4v14ceuo8E4PcLXhhGeDzCE7VM"
        }
      }
    ],
    "service": [
      {
        "id": "did:elsi:EU.EORI.NLPACKETDEL#info",

```



```

    "type": "EntityCommercialInfo",
    "serviceEndpoint": "https://packetdelivery.com/info",
    "name": "Packet Delivery co."
},
{
  "id": "did:elsi:EU.EORI.NLPACKETDEL#sms",
  "type": "SecureMessagingService",
  "serviceEndpoint": "https://packetdelivery.com/api"
}
],
"anchors": [
  {
    "id": "redt.alastria",
    "resolution": "UniversalResolver",
    "domain": "packetdelivery.ala",
    "ethereumAddress": "0xbcB9b29eeb28f36fd84f1CfF98C3F1887D831d78"
  }
],
"created": "2021-11-14T13:02:37Z",
"updated": "2021-11-14T13:02:37Z"
}
}

```

35. The DID Document includes one or more public keys inside the “verificationMethod” array. The keys are identified by the “id” field in each element of the array. The customer wallet uses the kid field that was received in the Authentication Request (in the protected header of the JWT) to select the corresponding Public Key and verify the signature of the JWT. It also verifies that the top-level “id” field in the DID Document (“did:elsi:EU.EORI.NLPACKETDEL”) is equal to the **client_id** parameter of the Authentication Request.
36. The customer wallet creates an Authentication Response to be posted in the **redirect_uri** specified by Packet Delivery company in step 5. The contents of the Authentication Response are described below.
37. The SIOP sends the authentication response to the endpoint passed in the **redirect_uri** authentication request parameter using a HTTP POST request using “application/x-www-form-urlencoded” encoding. The response contains an ID Token and a VP (Verifiable Presentation) token as defined in https://openid.net/specs/openid-connect-4-verifiable-presentations-1_0.html.

```

POST /siop_sessions HTTP/1.1
Host: client.example.com
Content-Type: application/x-www-form-urlencoded

id_token=eyJ0 ... NiJ9eyJ1c ... I6ljlifX0.DeWt4Qu ... ZXso

```



```
&vp_token=...
&state=af0ifjsldkj
```

The decoded **id_token** would be:

```
{
  "iss": "https://self-issued.me/v2",
  "aud": "did:elsi:EU.EORI.NLPACKETDEL",
  "iat": 1615910538,
  "exp": 1615911138,
  "sub": "did:peer:99ab5bca41bb45b78d242a46f0157b7d",
  "auth_time": 1615910535,
  "nonce": "n-0S6_WzA2Mj"
}
```

The **sub** claim is *did:peer:99ab5bca41bb45b78d242a46f0157b7d* which is the DID of the user and that is not registered in any blockchain or centralized repository. It must be the same as the DID included in the VP that was issued by the Happy Pets company when onboarding the customer and which travels in the authentication response.

The **vp_token** includes the Verifiable Presentation, which can be in two formats: **jwt_vp** (JWT encoded) or **Idp_vp** (JSON-LD encoded). The following example is using the JWT encoding:

```
{
  "format": "jwt_vp",
  "presentation": {
    "eyJhbGciOiJSUzI1NilsInR5cCl6IkpxVCIsImtpZCI6ImRpZDpleGFtcGxIOMfZmUxM2Y3MTIxMjA0
    MzFjMjc2ZTEyZWNhYiNrZXlzlTTeifQ.eyJzdWliOiJkaWQ6ZXhhbXBsZTpIYmZlYjFmNzEyZWJjNmYxY
    zl3NmUxMmVjMjEiLCJqdGkiOiJodHRwOi8vZXhhbXBsZS5lZHUVY3JIZGVudGlhbHMvMzczMilsImlzc
    yl6lmh0dHBzOi8vZXhhbXBsZS5jb20va2V5cy9mb28uandriwibmJmljoxNTQxNDkzNzI0LCJpYXQiO
    jE1NDE0OTM3MjQsImV4cCl6MTU3MzAyOTcyMywibm9uY2UiOi2NjAhNjM0NUZTZlilCJ2Yyl6eyJAY
    29udGV4dCl6WyJodHRwczovL3d3dy53My5vcmcvMjAxOC9jcmVkJZ50aWFscy92MSIsImh0dHBzOi8vd
    3d3LnczLm9yYz8yMDE4L2NyZWRlbnRpYWxzL2V4YW1wbGVzL3YxIl0sInR5cGUIOlsiVmVyaWZpYWJsZ
    UNyZWRlbnRpYWwiLCJvbml2ZXJzaXR5RGVncmVIQ3JIZGVudGlhbCJdLCJjcmVkJZ50aWFsU3ViamVjd
    CI6eyJkZWdyZWUiOnsidHlwZSI6IkjhY2hlbG9yRGVncmVIIiwibmFtZSI6ljxzcGFulGxhbm9J2ZyL
    UNBJz5CYWNjYWxhdXLdqWF0IGVuIG11c2lxWVzIG51bcOpclxdWVzPC9zcGFulGxhbm9J2ZyL
    BND3LDTn9H7FQokEsUEi8jKwXhGvoN3JtRa51xrNDgXD0c1UTYB-rK4Ft9YVmR1NI_ZOF8oGc_7wAp
    8PHbF2HaWodQloOBxxT-4WNqAxft7ET6IkH-4S6Ux3rSGAmczMohEEf8eCeN-jC8WekdPl6zKZQj0YPB
    1rx6X0-xIFBs7cl6Wt8rfBP_tZ9YgVWrQmUWypSioc0MUyiphmyEbLZagTyPiUyflGIEdqrZAv6eSe6R
    txJy6M1-ID7a5HTzanYTWBPAUHDZGyGKXdJw-W_x0IWChBzI8t3kpG253fg6V3tPgHeKXE94fz_QpYfg
    --7kLsyBAfQGbg"
  }
}
```



Which decoded could be:

```
{
  "@context": ["https://www.w3.org/2018/credentials/v1"],
  "type": ["VerifiablePresentation"],
  "verifiableCredential": [
    {
      "@context": [
        "https://www.w3.org/ns/credentials/v2",
        "https://happypets.fiware.io/2022/credentials/employee/v1"
      ],
      "id": "https://happypets.fiware.io/credentials/25159389-8dd17b796ac0",
      "type": ["VerifiableCredential", "CustomerCredential"],
      "issuer": {
        "id": "did:elsi:EU.EORI.NLHAPPYPETS"
      },
      "issuanceDate": "2022-03-22T14:00:00Z",
      "validFrom": "2022-03-22T14:00:00Z",
      "expirationDate": "2023-03-22T14:00:00Z",
      "credentialSubject": {
        "id": "did:peer:99ab5bca41bb45b78d242a46f0157b7d",
        "verificationMethod": [
          {
            "id": "did:peer:99ab5bca41bb45b78d242a46f0157b7d#key1",
            "type": "JwsVerificationKey2020",
            "controller": "did:peer:99ab5bca41bb45b78d242a46f0157b7d",
            "publicKeyJwk": {
              "kid": "key1",
              "kty": "EC",
              "crv": "P-256",
              "x": "IJtvoA5_XptBvcfcrtGCvXd9bLymmfBSSdNJf5mogo",
              "y": "fSc4gZX2R3QKKfHvS3m2vGSVSN8Xc04qsquyfEM55Z0"
            }
          }
        ],
        "roles": [
          {
            "target": "did:elsi:EU.EORI.NLPACKETDEL",
            "names": ["P.Info.gold"] // Or P.Info.standard
          }
        ],
        "name": "Jane Doe",
        "given_name": "Jane",
        "family_name": "Doe",
        "preferred_username": "j.doe",
        "email": "janedoe@packetdelivery.com"
      }
    }
  ]
}
```



```
}
```

38. Packet Delivery company uses its own blockchain node implementing the Universal Resolver functionality to resolve the DID of Happy Pets, which is inside the Verifiable Credential received in the Verifiable Presentation. This DID can be found in the “issuer” field of the “verifiableCredential” structure above.

Resolution is performed sending a GET request to the Universal Resolver:
/api/did/v1/identifiers/did:elsi:EU.EORI.NLHAPPYPETS

Packet Delivery could use a Universal Resolver operated by a different entity, but this would reduce the level of trust compared to using its own server directly connected to the blockchain network.

39. Packet Delivery receives the DID Document of Happy Pets with trusted information about the company, including the Public Key associated to the Private Key that Happy Pets used to digitally sign the Verifiable Credential that the customer has just sent inside a Verifiable Presentation as part of the authentication flow. **Using the Public Key and the DID inside the DID Document, it can verify the signature of the Verifiable Credential and that Happy Pets is a trusted entity in the ecosystem.**
40. The above is just for verification of the Verifiable Credential. In addition, Packet Delivery company can also verify that the Verifiable Presentation including the Verifiable Credential is sent by the customer and not by a malicious agent. To do so, it uses the Public Key of the customer in the “verificationMethod” of the “credentialSubject” structure. That public key is cryptographically bound to the customer DID during the onboarding process that Happy Pets performed with the customer.
41. Once all verifications have been performed, Packet Delivery company creates an Access Token for the customer so she can use it to access services in Packet Delivery company in the future.
42. The wallet (SIOP) receives a successful reply to the POST request.

43. The Packet Delivery company proxy notifies the Packet Delivery portal that the customer is successfully authenticated, and the portal can display the services available to that customer. The browser of the user receives the Access Token created by Packet Delivery to enable it to request services without going through the previous authentication process. The Access Token is a standard OAuth access token that includes the information that Packet Delivery requires for accessing its services.

At this point the Happy Pets customer is logged in on the Packet Delivery company portal/app and is presented with the possible services provided, including the option to change the PTA of its delivery orders.



At this moment, the Packet Delivery company knows the following:

- Happy Pets is a participant in the Data Space and that it is a Trusted Issuer of EmployeeCredentials because this info is in the DID Document retrieved in step 13. Maintenance of this information is performed by the Trusted Anchor entity(or entities) managing the Trusted Participants List and Trusted Issuers Registry.
- Happy Pets says that the user is a customer. This info is inside the Verifiable Credential that is digitally signed by Happy Pets.
- The category of the customer (and associated policies) with regards to the services offered by Packet Delivery company. This information is also in the Verifiable Credential presented by the customer.

44. The Happy Pets customer is presented with the possible services provided by Packet Delivery, including the option to change the PTA of its delivery orders.
45. Happy Pets customer searches for his packet delivery order and is presented its details. He now requests a change of the PTA of this order on the Packet Delivery company portal/app.
46. Packet Delivery company portal/app sends a request to Packet Delivery company proxy, in order to change the PTA of the delivery order. The request contains the Access Token generated in step 15, with information about the authorisation registry to retrieve policies from.

```
> Authorization: Bearer lIeD...NIQ // Bearer JWT
> Content-Type: application/json
```

PATCH <https://umbrella.fiware.io/ngsi-ld/v1/entities/urn:ngsi-ld:DELIVERYORDER:001/attrs/pta>

```
> Payload
{
  "value": "<new PTA>",
  "type": "Property"
}
```

Decoded Bearer JWT payload:

```
{
  "iss": "EU.EORI.NLHAPPYPETS", // Issuer: Happy Pets
  "sub": "419404e1-07ce-4d80-9e8a-eca94vde0003de", // Customer pseudonym
  "jti": "d8a7fd7465754a4a9117ee28f5b7fb60",
  "iat": 1591966224,
  "exp": 1591966254,
  "aud": "EU.EORI.NLHAPPYPETS",
  "authorisationRegistry": { // AR to retrieve policies from
    "url": "https://ar.packetdelivery.com",
    "identifier": "EU.EORI.NLHAPPYPETS",
```



```

    "delegation_endpoint": "https://ar.packetdelivery.com/delegation",
}
}

```

47. Packet Delivery company proxy received the request of step 19 for changing the PTA of a delivery order. The Access Token received from the customer ensures that she was assigned the delegation evidence with a policy for updating the PTA attribute of this specific delivery order (called issuance at **user level**). Furthermore, since in this scenario the required customer policy was issued by a 3rd party (Happy Pets), the proxy has to check whether Happy Pets itself is allowed to delegate this policy. In general, the rule would be that the proxy needs to check the existence of valid policies through the chain of issuers, until itself (in this case the Packet Delivery company) is the issuer. In this scenario, the proxy will check policies at two different levels: issued at **organizational level** (from Packet Delivery company to Happy Pets) and issued at **user level** (from Happy Pets to customer). The Verifiable Credential takes care of the user level policies.

At first, the Packet Delivery company proxy validates the JWT which is part of the authorization header of the PATCH request.

48. In order to check whether Happy Pets is allowed to delegate the policy to its customers, the proxy will check at the Packet Delivery company Authorisation Registry whether this policy exists. The proxy sends a request to the /delegation endpoint of the Packet Delivery company Authorization Registry.

49. The proxy receives the delegation evidence policy issued from Packet Delivery company to Happy Pets.

50. Having received the delegation information from the Packet Delivery company Authorization Registry, the proxy (or more precisely, the PDP) can now evaluate whether the contained **organisational policy** allows for updating the PTA attribute, and therefore whether Happy Pets is allowed to delegate the access to its customers. If the proxy received a valid policy, access would be granted on an **organisational level**.

If the requested delegation evidence can not be found or the returned policy contains the Deny rule, the change of the PTA would be denied by the Packet Delivery company proxy and an error would be returned to the Packet Delivery company portal/app, also presented to the Happy Pets customer. The following steps would be omitted.

51. As described in the previous steps, the PDP evaluated that a change of the PTA of the specific delivery order is granted, both on **organisational level** and **user level**. As a result, the request for changing the PTA is forwarded by the Packet Delivery company



proxy to the Packet Delivery company Context Broker which holds the information of the packet delivery order. The PTA of the packet delivery order is changed and the Context Broker returns a successful response with HTTP code 204. The Context Broker response is returned to the Packet Delivery company portal, in response to the request of step 26.

52. The successful change of the PTA is presented to the Happy Pets customer.

There are some variations of above steps in the scenario of the No Cheaper customer.

Basically the sequence of steps is the same as for Happy Pets. In contrast to Happy Pets, during the acquisition of rights described in [3.5.3.3.2 Acquisition of Rights / Activation](#), No Cheaper is just acquiring the standard service and therefore its customers will only be able to read attributes of delivery orders. This means that at the Packet Delivery authorisation registry, there is only a policy created allowing No Cheaper to only delegate GET access to delivery orders.

This scenario can be split into two cases to demonstrate the denial of access based on the different policies on organisational level and user level.

1. At No Cheaper Authorisation Registry, a Verifiable Credential is issued to the No Cheaper customer allowing only GET requests to the Packet Delivery service (representing the P.Info.Standard role). When performing the steps for changing the PTA value of a delivery order, as described in the previous section, the process would stop at step 43, where access would be rejected because the No Cheaper customer was not assigned the necessary policy at user level.
2. At No Cheaper Authorisation Registry, a Verifiable Credential is issued to the No Cheaper customer allowing both GET and PATCH requests to the Packet Delivery service (representing the P.Info.Gold role). When performing the steps for changing the PTA value of a delivery order, as described in the previous section, the process would stop at step 62, where access would be rejected because No Cheaper was not assigned the necessary policy at the Packet Delivery company Authorisation Registry to delegate the premium access to its customers. Therefore access would be rejected at organisational level. This is to show that access would be still rejected, even when the No Cheaper organisation issues access to the premium service to its customers within its own Authorization Registry.

In general, for both cases the request for changing the PTA should be denied. However, it can be shown that the No Cheaper customer is able to view attributes of its delivery orders.

3.4.3.3.4 Issuing tokens for Connectors / application context

In addition to the section described above, tokens (DAT Dynamic Access Token) for the application context must be issued containing the referencing connectors and their security profile. The details of issuing the tokens have to be described to act as an alternative to the current IDS-DAPS realisation or to include these mechanisms as specified in [IDS-G](#).



The DAPS issues the requested DAT, or an error response, as per RFC 6749. The Access Token ("the DAT") itself is a JWS adhering to RFC 9068, which in turn contains JSON-LD encoded data in addition to the standard claims, subject to the following additional constraints:

Field name	additional constraints
@context	Must be https://w3id.org/idsa/contexts/context.jsonld
@type	Must be ids:DatPayload
securityProfile	Must be an instance of the ids:SecurityProfile class

The DAT MUST be signed using a digital signature scheme. It SHOULD be limited to a short time period (Recommendation: 1 hour). The default resource indicator to be used in the DAT includes idsc:IDS_CONNECTORS_ALL, which SHOULD be accepted by all connectors. Future revisions of this document may allow for mechanisms to specify connectors to be listed in the aud claim such as through RFC 8707.

Additional claims may optionally be present. This specification defines the following:

- **referringConnector** An optional URI of the subject. Is used to connect the identifier of the connector with the self-description identifier as defined by the IDS Information Model. A receiving connector can use this information to request more information at a Broker or directly by dereferencing this URI.
- **transportCertsSha256** Contains the public keys of the used transport certificates, hashed using SHA256. The identifying X509 certificate should not be used for the communication encryption. Therefore, the receiving party needs to connect the identity of a connector by relating its hostname (from the communication encryption layer) and the used private/public key pair, with its IDS identity claim of the DAT. The public transportation key must be one of the transportCertsSha256 values. Otherwise, the receiving connector must expect that the requesting connector is using a false identity claim. In general, this claim holds an Array of Strings, but it may optionally hold a single String instead if the Array would have exactly one element.
- **extendedGuarantee** In case a connector fulfils a certain security profile but deviates for a subset of attributes, it can inform the receiving connector about its actual security features. This can only happen if a connector reaches a higher level for a certain security attribute than the actual reached certification asks for. A deviation to lower levels is not possible, as this would directly invalidate the complete certification level. In general, this claim holds an Array of Strings, but it may optionally hold a single String instead if the Array would have exactly one element.

Example

The following is an example of a successful response:



200 This is fine

Content-Type: application/json

```
{  
  "access_token": "skdj54dkGjnb[...]lsl8723ijfdfuzticby_ch",  
  "scope": "idsc:IDS_CONNECTOR_ATTRIBUTES_ALL",  
  "token_type": "bearer",  
  "expires_in": "3600"  
}
```

The decoded DAT, including header and payload is shown below:

```
{  
  "typ": "jwt+at",  
  "kid": "somekid",  
  "alg": "RS256"  
}  
. . .  
{  
  "iss": "https://daps.aiese.fraunhofer.de/v3",  
  "sub":  
"DD:CB:FD:0B:93:84:33:01:11:EB:5D:94:94:88:BE:78:7D:57:FC:4A:keyid:CB:8C:C7:B6:85:  
79:A8:23:A6:CB:15:AB:17:50:2F:E6:65:43:5D:E8",  
  "nbf": 1516239022,  
  "iat": 1516239022,  
  "exp": 1516239032,  
  "aud": ["idsc:IDS_CONNECTORS_ALL"],  
  "scope": "idsc:IDS_CONNECTOR_ATTRIBUTES_ALL",  
  "@context": "https://w3id.org/idsa/context/context.jsonld",  
  "@type": "ids:DatPayload",  
  "referringConnector": "http://some-connector-uri.com",  
  "securityProfile": "idsc:BASE_SECURITY_PROFILE",  
  "extendedGuarantee": "idsc:USAGE_CONTROL_POLICY_ENFORCEMENT",  
  "transportCertsSha256": "bacb879575730bb083f283fd5b67a8cb..."  
}  
. . .  
somesignature
```



4 DOME Marketplace Persistence Layer

4.1 Introduction

4.1.1 Document Objective

This document aims to provide an overview and an outline of the objectives of the DOME Persistence Layer, focusing on the specific goals and priorities associated with its implementation.

The main goal of the DOME Persistence Layer is to provide the necessary persistence and storage capabilities for the two conceptual components: the *Shared Catalog*, and the *Transaction Ledger*.

- The **Shared Catalog** stores descriptions and specifications (Product Specification and Product Offerings) of cloud and edge services in the form of Verifiable Credentials and/or Verifiable Presentations, and other information for the DOME ecosystem consumption.
- The **Transaction Ledger** securely records transactions, including products (the instances of the Product Specifications), Product Orders, and Product Usage.

4.1.2 Context and Rationale for Adopting a Decentralised and Federated Data Architecture in the DOME Solution Ecosystem

In the context of the DOME ecosystem where each Participant has its own solution, the problem of data persistence becomes increasingly complex due to data silos and the lack of integration. To overcome these challenges and unlock the true potential of data within the ecosystem, the adoption of a Decentralised Persistence Layer (DPL) emerges as a highly effective solution.

1. Breaking Down Data Silos
2. Ensuring Data Consistency and Integrity
3. Enhanced Data Visibility and Accessibility
4. Simplified Data Integration
5. Robust Security and Compliance



6. Scalability and Flexibility

However, while a DPL offers significant benefits, it is essential to consider other critical aspects alongside its implementation.

One such aspect is ensuring compliance with data protection regulations, like the General Data Protection Regulation (GDPR). DOME must carefully design the DPL to adhere to privacy and security requirements, enabling proper data handling, consent management, and the ability to enforce data subject rights. In any case, no PII (personally identifiable information) data will ever be stored in a global database or registry like a blockchain. This type of date will always be stored off-chain and with proper management procedures.

Additionally, it's important to recognize that certain data may still reside in off-chain databases or legacy systems. Co-existing with off-chain databases necessitates effective data synchronisation mechanisms to ensure consistency and accuracy across the ecosystem.

4.2 Theoretical Foundations

4.2.1 Decentralised Persistent Layer

4.2.1.1 Definition

A Decentralised Persistence Layer (**DPL**) refers to a technological component or infrastructure that provides persistence and storage capabilities in a decentralised manner. It is designed to store and manage data in a distributed fashion, usually across multiple interconnected networks or blockchains.

The DPL serves as a common layer that decouples data storage and management from individual applications, creating a unified and shared data environment. It facilitates the seamless flow of data, breaking down data silos and enabling efficient data integration and collaboration among various stakeholders.

The main characteristics of a Decentralised Persistent Layer include:

1. Data Distribution
2. Data Consistency
3. Standardised APIs
4. Security and Access Control
5. Scalability and Flexibility

4.2.1.2 Decentralization in Data Storage



Decentralisation in Data Storage implies that data is spread across multiple storage locations or nodes (Access Nodes).

This Decentralised Architecture offers several advantages over traditional Centralised Architectures. Implementing the Decentralised Architecture we can provide increased resilience, scalability, security, and data availability.

In the Decentralised Architecture, data is fragmented and distributed across multiple nodes (other Access Nodes instances). Each Access Node in the ecosystem holds a portion of the data, and together they form a distributed network.

The key characteristics of decentralisation in data storage include:

1. Redundancy and Fault Tolerance
2. Increased Data Availability
3. Scalability
4. Enhanced Security
5. Data Sovereignty and Privacy
6. Data Consistency and Integrity
7. Peer-to-Peer Collaboration

4.2.1.3 Blockchain and Context Broker as a Technology Solution

In the development of the distributed persistence layer for the DOME solution ecosystem, careful consideration was given to selecting the appropriate technology solutions to ensure reliable and efficient data management.

The combination of blockchain technology and the Context Broker emerged as a powerful and synergistic approach to meet the specific requirements of the system. Together, blockchain technology and the Context Broker form a cohesive technology solution for the distributed persistence layer in the DOME ecosystem.

This combination provides a secure, transparent, and scalable infrastructure for storing and managing transactional data on the blockchain while efficiently managing full entity information off-chain.

4.2.1.3.1 Blockchain Technology for Distributed Persistence

Blockchain technology was chosen as a fundamental component of the distributed persistence layer due to its inherent characteristics that align with the goals of decentralisation, immutability, and transparency. Furthermore, it provides a secure and decentralised ledger for recording transactions and events.



One of the key advantages of blockchain is its ability to publish events with partial information. This feature allows the Distributed Persistence Layer to capture and store relevant information about transactions, such as Product Specifications, Product Offerings, Product Orders, and Product Usage data, while maintaining the necessary privacy and security of sensitive information.

The use of blockchain technology ensures the DPL to the integrity and immutability of the recorded data because once a transaction is recorded on the blockchain, it becomes tamper-proof and resistant to unauthorised modifications. This property enhances trust among DOME participants and provides a reliable source of truth for auditing and verification purposes.

4.2.1.3.2 Context Broker as an Off-chain Persistence Solution

While blockchain technology offers strong guarantees of data integrity, it may not be suitable for storing large amounts of detailed entity information due to scalability limitations. To address this challenge, the Context Broker is employed as an off-chain persistence solution within the distributed persistence layer.

The Context Broker serves as a complementary component that provides full entity information storage. It allows for the efficient storage and retrieval of detailed information related to entities.

By utilising the Context Broker for off-chain persistence, the distributed persistence layer can ensure efficient storage and access to entity information while benefiting from the scalability and performance advantages it offers. This symbiotic relationship between blockchain technology and the Context Broker enables a balance between data integrity, privacy, and efficient data management.

4.2.1.4 Decentralised Networks

In the digital age, traditional centralised systems have been the norm for many applications, but they come with inherent drawbacks such as single points of failure, limited scalability, and susceptibility to censorship and data breaches. Decentralised networks, on the other hand, offer a revolutionary approach to address these issues. By distributing control, storage, and processing across a network of nodes, decentralised networks offer enhanced security, resilience, and transparency. The document will delve into the technology behind decentralised networks, focusing on key concepts such as Network Nodes and Distributed Storage, Replication and Consensus Mechanisms, and Data Security and Encryption.

4.2.1.4.1 Network Nodes and Distributed Storage

In a decentralised network, Network Nodes play a pivotal role. A network node refers to an individual device or computer that participates in the network and performs various tasks. These tasks can include storing and validating data, processing transactions, maintaining the network's consensus, and propagating information to other nodes.

The distribution of nodes in a decentralised network ensures redundancy and fault tolerance. Unlike traditional centralised systems, where a single server outage can bring down an entire service, decentralised networks can continue working as long as there is a sufficient number of operational nodes. This inherent redundancy makes decentralised networks more robust and resilient to failures or attacks.

Distributed Storage is another fundamental aspect of decentralised networks. Instead of relying on a single central server to store data, decentralised networks distribute data across



multiple nodes. This approach, often known as Distributed Ledger Technology (DLT), ensures that data is replicated across various nodes in a secure and transparent manner. Distributed storage mechanisms, such as blockchain, allow all nodes in the network to maintain a synchronised and tamper-resistant copy of the data.

4.2.1.4.2 Replication and Consensus Mechanisms

Replication is a critical aspect of decentralised networks that ensures data redundancy and availability. When data is replicated, it is copied across multiple nodes within the network. This replication process helps prevent data loss, increases data availability, and contributes to the overall fault tolerance of the system.

Consensus mechanisms are essential for maintaining agreement among nodes in a decentralised network. As multiple nodes can participate in the network and propose changes, consensus mechanisms ensure that all nodes reach agreement on the state of the network and the validity of transactions. By achieving consensus, decentralised networks avoid issues like double-spending and data inconsistencies that can arise in distributed environments.

Consensus protocols are fundamental to the functioning of decentralised networks, ensuring agreement among network participants regarding the validity of transactions and the state of the blockchain. Different consensus protocols have been developed, each with its own set of advantages and disadvantages. In this document, we will explore and compare the following consensus protocols: Proof of Work (PoW), Proof of Stake (PoS), Delegated Proof of Stake (DPoS), Improved Finality Byzantine Fault Tolerance (IFBT), Practical Byzantine Fault Tolerance (PFBT), and Quorum-Based Practical Byzantine Fault Tolerance (QFBT).

Proof of Work (PoW)

Advantages:

- **Security:** PoW is highly secure due to the resource-intensive nature of mining, making it difficult and costly for attackers to control the network.
- **Decentralisation:** PoW can be decentralised at the beginning, but with time, networks become highly centralised systems because only powerful entities can have the computational power to have an acceptable probability of winning the cryptographic lottery essential for PoW.

Disadvantages:

- **Energy Consumption:** PoW is notorious for its high energy consumption, leading to environmental concerns.
- **Centralization of Mining Power:** Over time, mining has become centralised in the hands of large mining pools, vastly reducing the decentralisation aspect.

Proof of Stake (PoS)

Advantages:

- Energy Efficiency: PoS consumes significantly less energy compared to PoW, making it a more sustainable option.
- Decentralisation Incentive: PoS encourages participants to hold and stake their cryptocurrency. It promotes centralisation because it promotes the “rich gets richer” paradigm. Together with PoW, PoS belongs to the family of consensus algorithms known as PoR (Proof of Richness).

Disadvantages:



- **Wealth Concentration:** PoS can lead to wealth concentration, as those with more cryptocurrency have a higher chance of being chosen as validators.
- **Nothing at Stake:** In theory, PoS validators can support multiple forks without suffering consequences, which could lead to potential issues.

Delegated Proof of Stake (DPoS)

Advantages:

- **Efficiency:** DPoS is faster and more efficient than PoW and PoS, allowing for quicker transaction confirmation times.
- **Sybil Resistance:** DPoS introduces the concept of voting for delegates, making it harder for attackers to create multiple identities (Sybil attacks).

Disadvantages:

- **Centralization Risk:** DPoS relies on a limited number of elected delegates, which could lead to centralization if the same delegates are continually chosen.
- **Voter Apathy:** Token holders might not actively participate in voting, leading to a concentration of power in the hands of a few participants.

Improved Finality Byzantine Fault Tolerance (IFBT)

Advantages:

- **Finality:** IFBT achieves faster finality, ensuring that confirmed transactions cannot be reverted or changed.
- **Low Resource Consumption:** IFBT is more resource-efficient compared to PoW and traditional BFT.

Disadvantages:

- **Limited Scalability:** IFBT may face challenges in scaling to handle a large number of transactions and participants.
- **Network Partitioning:** In case of network partitioning, some nodes might be unable to reach consensus, potentially leading to disruptions.

Practical Byzantine Fault Tolerance (PFBT)

Advantages:

- **Finality and Low Latency:** PFBT provides fast transaction finality and low latency for confirming transactions.
- **Fault Tolerance:** PFBT can handle a certain number of faulty nodes while still maintaining consensus.

Disadvantages:

- **Limited Decentralisation:** PFBT might have limitations on the number of participating nodes, potentially impacting decentralisation.
- **Complexity:** Implementing PFBT can be more complex than other consensus protocols.

Quorum-Based Practical Byzantine Fault Tolerance (QFBT)

Advantages:



- **Flexible Quorum Selection:** QFBT allows for flexibility in selecting the quorum size and participants, offering adaptability.
- **High Throughput:** QFBT can achieve high throughput, making it suitable for applications with large transaction volumes.

Disadvantages:

- **Quorum Management:** Proper quorum management is essential, as the wrong quorum selection could lead to security vulnerabilities.
- **Synchronisation Delay:** Network synchronisation delays might impact QFBT's performance and efficiency. In any case, it is always much better than PoW or PoS.

Consensus Protocol	Advantages	Disadvantages
Proof of Work (PoW)	- Security	- Energy Consumption
	- Decentralisation in early stages of the network	- Centralization of Mining Power as the network matures
Proof of Stake (PoS)	- Energy Efficiency	- Wealth Concentration
	- Decentralization Incentive (only for entities that can stake).	- Nothing at Stake
Delegated Proof of Stake (DPoS)	- Efficiency	- Centralization Risk



	- Sybil Resistance	- Voter Apathy
Improved Finality Byzantine Fault Tolerance (IFBT)	- Finality	- Limited Scalability
	- Low Resource Consumption	- Network Partitioning
Practical Byzantine Fault Tolerance (PFBT)	- Finality and Low Latency	- Limited Decentralisation
	- Fault Tolerance	- Complexity
Quorum-Based Practical Byzantine Fault Tolerance (QFBT)	- Flexible Quorum Selection	- Quorum Management
	- High Throughput	- Synchronisation Delay

In conclusion, each consensus protocol has its own strengths and weaknesses, and the choice of which one to use depends on the specific requirements of the decentralised network and the trade-offs that the network developers are willing to make. Factors such as security, energy efficiency, decentralisation, and scalability play crucial roles in determining the most suitable consensus protocol for a given blockchain application.



4.2.1.4.3 Data Security and Encryption

Data security is a paramount concern in decentralised networks. Since data is distributed across multiple nodes, ensuring the confidentiality, integrity, and authenticity of data is essential.

Encryption plays a pivotal role in data security. Data transmitted between nodes and stored on the network is typically encrypted to prevent unauthorised access. Encryption techniques such as public-key cryptography enable secure communication and transactions within the decentralised network.

In addition to encryption, decentralisation contributes to data security by reducing the risk of single points of failure and data breaches. Traditional centralised systems are attractive targets for hackers because compromising a central server could grant access to vast amounts of data. Decentralised networks, on the other hand, require attackers to compromise multiple nodes, making such attacks significantly more challenging and less rewarding.

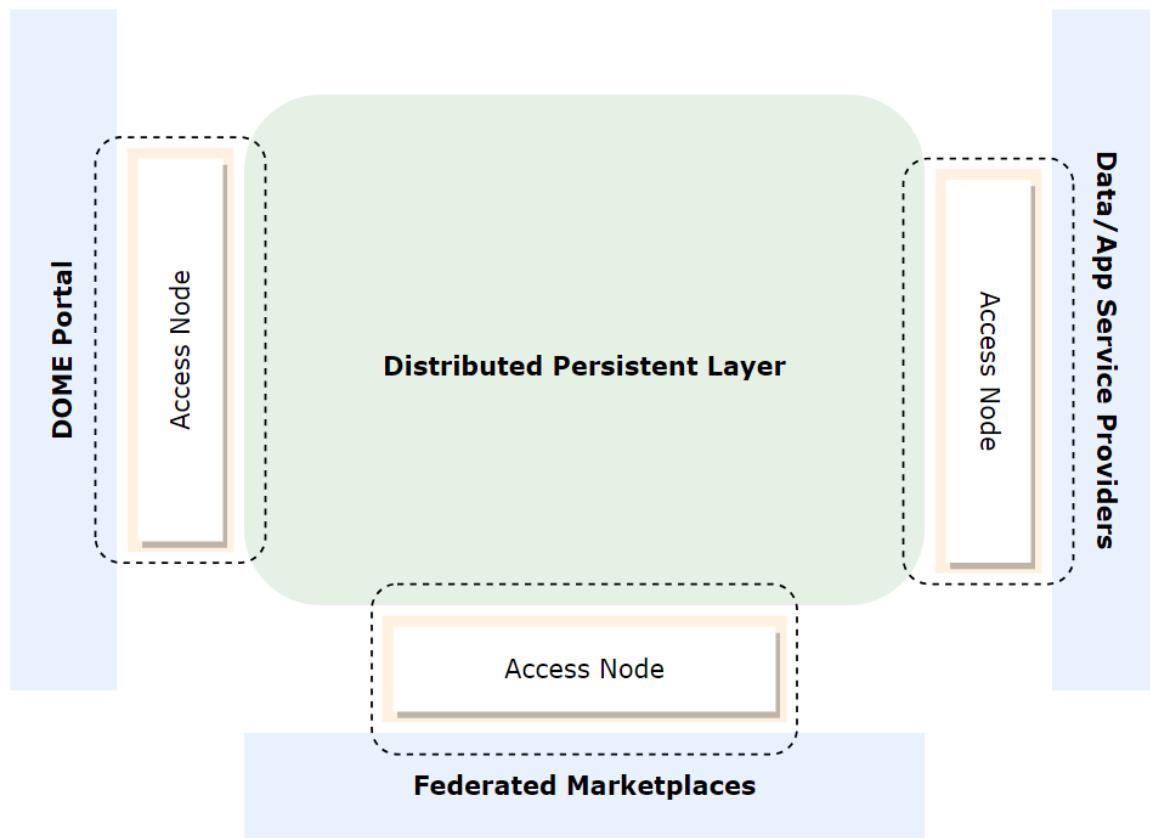
To maintain data security, decentralised networks often utilise robust authentication and access control mechanisms. Nodes must prove their identity before participating in the network, and permissions are carefully managed to ensure that only authorised nodes can perform specific actions.

4.3 Proposed Architecture

4.3.1 Overview of the Architecture

Within the Distributed Persistence Layer, a conceptual division is established based on the nature of the data: the **Shared Catalog** and the **Transaction Ledger**.





Let's delve into each of these concepts and their roles within the Distributed Persistence Layer

1. Shared Catalog

The Shared Catalog is responsible for storing Product Specifications (including the specifications of associated services and supporting resources), Product Offerings defined by service providers.

2. Transaction Ledger

The Transaction Ledger is a crucial component of the DOME Persistent Layer. It serves as a record-keeping mechanism for all transactional activities within the DOME environment. That is, Product Orders and Product (instances of Product Specifications) along their lifecycle, as well as information about actual Usage of Products.

In the DOME architecture, both the Shared Catalog and Transaction Ledger are utilised by various participants, including the **DOME Portal Back-End**, **Data/App Service Provider Back-Ends**, and **Federated Marketplace Back-Ends**. These participants access the DOME Persistent Layer through an **Access Node** component.



The Access Node acts as an interface for participants to access the DOME Persistent Layer. It consists of three main components: the **TM Forum APIs** component, the **Storage Back-End** component, and the **Blockchain Connector** component.

- The **TM Forum API** component provides standardised entry points for accessing the DOME Persistent Layer.
- The **Storage Back-End** component serves as the underlying off-chain storage infrastructure.
- The **Blockchain Connector** component facilitates the integration between the participant back-ends and the blockchain, ensuring the synchronisation of relevant data and events.

4.3.1.1 Blockchain Federation Architecture

The DOME Decentralised Persistence Layer will be implemented on top of a number of interconnected national blockchains compatible with the European Blockchain Service Infrastructure (EBSI) when not directly EBSI.

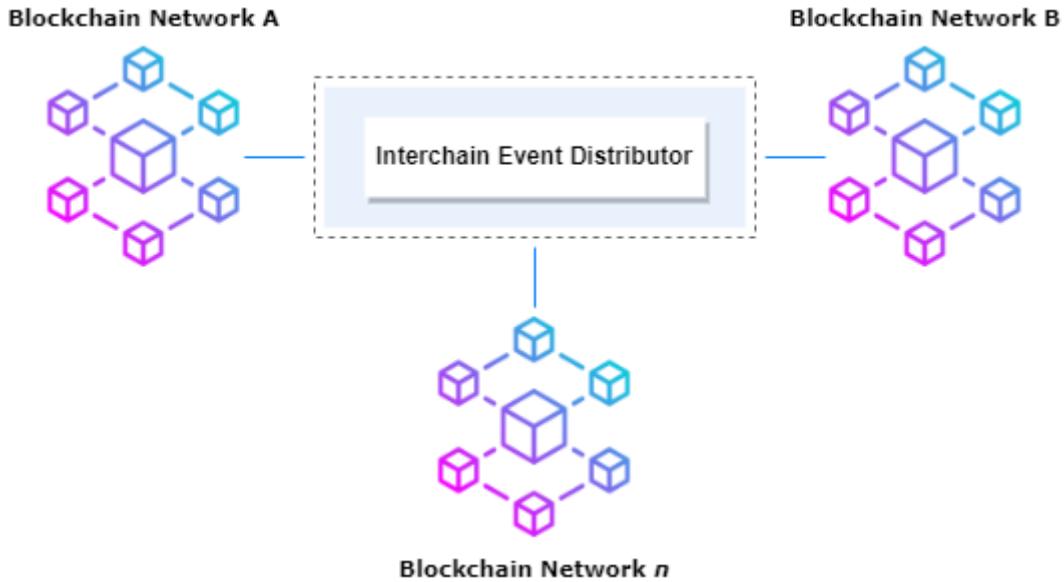
In this architecture, multiple independent blockchain networks are interconnected and communicate with each other through an Interchain Event Distributor component.

At its core, the Blockchain Federation Architecture aims to combine the benefits of multiple blockchain networks while maintaining their autonomy and preserving their individual consensus mechanisms. The DOME Distributed Persistence Layer Architecture achieves this by establishing a network of interconnected blockchain networks.

The key component that enables communication between the federated blockchain networks is the **Interchain Event Distributor (IED)**. This component facilitates the seamless exchange of events, transactions, and other relevant data between the participating blockchain networks. By leveraging event-based communication, the DPL architecture ensures timely and efficient synchronisation of information across the federated network.

Each blockchain network within the federation retains its own conceptual architecture, including its unique consensus mechanisms, data storage, and smart contract functionalities. This distributed approach allows for flexibility and scalability, as new blockchain networks can be added to the federation seamlessly.





4.3.2 Components of the DOME Decentralised Persistence Layer

4.3.2.1 TM Forum APIs

TM Forum APIs should provide a standardised interface for seamless integration and interoperability across different systems and solutions.

They are a suite of application programming interfaces designed to facilitate the management of services throughout their lifecycle in a collaborative environment involving multiple partners.

These APIs provide a standardised and efficient approach to managing various services, allowing for seamless integration, interoperability, and simplified service management.

By adhering to industry-strength design patterns, the APIs enable rapid implementation and consistent management of services.

Based on representational state transfer (REST), the TM Forum APIs are technology-agnostic, making them applicable to a wide range of digital service scenarios. These scenarios can include B2B interactions, Internet of Things (IoT) deployments, Smart Health applications, Smart Grid solutions, Big Data analytics, Network Function Virtualization (NFV), Next Generation OSS/BSS (Operations Support Systems/Business Support Systems), and more.

These characteristics will be explained in detail in section 6, **DOME marketplace features**, of this document.

4.3.2.2 Storage Back-End

The data in the origin organisation can be stored in an existing system or in a new one. The only requirement is that the data can be accessed by other organisations in the network using standardised pointers (URIs), APIs, access control and with a data model which is the same for all organisations in the network.



The handling of personal data has significant legal implications, as it involves sensitive information that must be protected to ensure privacy and confidentiality. Personal data is subject to various legal requirements, such as data minimization, consent, and the right to be forgotten, among others. These legal requirements are in place to ensure that personal data is collected and processed in a responsible and ethical manner, and that individuals have control over their personal information. As a result, stringent measures must be put in place to safeguard personal data, as it carries legal, ethical, and moral obligations.

When data is stored, it can be done so by the owner of the data or by a third-party entity that stores it on behalf of the owner. When it comes to personal data, special care must be taken to ensure that the data is stored securely and in compliance with legal regulations. In some cases, users may store their own personal data on their own devices, such as mobile phones or hard disks.

However, a critical situation arises when a legal entity such as a business or government stores and manages personal data of its customers or citizens. Depending on the specific circumstances, this entity may be considered a Data Controller or a Data Processor. In either case, the entity is subject to strict legal requirements under GDPR (General Data Protection Regulation) that govern the handling, storage, and processing of personal data.

Data Controllers have primary responsibility for compliance with GDPR and are obligated to ensure that personal data is collected and processed in a legal and ethical manner. Data Processors, on the other hand, are obligated to process personal data only on behalf of the Data Controller and in compliance with GDPR. Regardless of whether an entity is a Data Controller or a Data Processor, it is subject to the stringent legal requirements set forth by GDPR, and must take all necessary steps to ensure the safety and confidentiality of personal data.

Efficiently storing and transferring large amounts of data can be a complex task compared to smaller amounts of data. While it is challenging to define a precise threshold for what constitutes a “small” or “large” amount of data, generally, small amounts of data can be transferred using a “push” or “dissemination” model, whereas large amounts of data are better transferred using a “pull” or “on-demand” model.

With a push or dissemination model, data is sent to the recipient without any specific request. This model works well for smaller amounts of data since it is quick and straightforward. However, when it comes to larger amounts of data, this model can be inefficient, as it may take a lot of time to transfer and can consume a significant amount of network bandwidth.

On the other hand, the pull or on-demand model is better suited for transferring large amounts of data. In this model, data is only transferred when it is specifically requested by the recipient, reducing the amount of unnecessary data transfer. This model can be more efficient for larger amounts of data, as it allows for more targeted and precise data transfers.



4.3.2.2.1 Standardised interfaces and mechanisms

To ensure seamless interoperability among all members of DOME, it is essential that all participants use the same standard mechanisms and APIs for transferring and accessing data, regardless of the underlying data storage systems used in each organisation. At the technical layer of DOME, several key aspects need to be addressed:

1. **A safe pointer to the origin data is required.** This will enable data to be tracked and located accurately throughout the system.
2. **A standard API for data access operations is needed.** This API should be independent of any specific database product or technology, allowing for greater flexibility and interoperability.
3. **A common metamodel and data representation, such as JSON-LD, should be adopted** to ensure semantic interoperability and data linking.
4. **A common access control mechanism must be established to restrict data access** to authorised organisations in accordance with a given policy. This could include restrictions based on one-time access, time-based access, or other relevant factors.

Based on this common metamodel and data representation, each use case can build their own specific data models, which will be domain-specific and not described in this document.

To implement these aspects in DOME, it is recommended that each participant uses the same standard mechanisms and APIs for data transfer and access. This will ensure that all data is accessible and transferable between organisations in a seamless and efficient manner, regardless of the underlying storage systems used.

W3C in the paper [A JSON-based Serialization for Linked Data](#) defines that JSON-LD is a lightweight syntax used to serialise Linked Data in JSON, allowing interoperability and reuse of existing JSON libraries. It is commonly used in web-based programming environments to build interoperable web services and store Linked Data in JSON-based storage engines.

[CEF Context Broker](#) CEF Context Broker is a platform that enables organisations to manage and share real-time data, such as traffic status, air quality, and parking availability. The platform leverages the power of [Linked Data](#) to create consistent models for data coming from different sources, using a standard API and information model. The information model is defined at two levels: the foundation classes, which correspond to the Core Metamodel, and the Cross-Domain Ontology. The [ETSI NGSI-LD API](#) is being extended and standardised for use with the CEF Context Broker, providing a common framework for managing and sharing data.

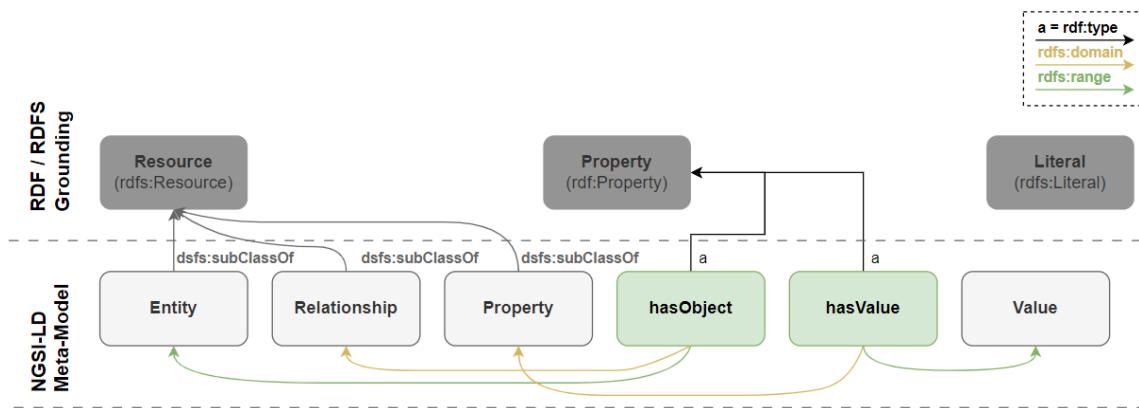
4.3.2.2.2 CEF Context Broker reference implementation.

The ETSI NGSI-LD API can be implemented on top of different storage mechanisms and databases. For DOME a good option is to use the reference Open Source implementation which has been already used in production in different systems, most notably in Smart Cities



and currently being expanded to other use cases like Smart Industry. The Orion Context Broker is such an implementation, and this is its [GitHub](#) link.

The CEF Context Broker API leverages the power of Linked Data to facilitate semantic interoperability of the data exchanges across the different organisations. The core metamodel is described in the following diagram:



4.3.2.2.3 Connecting Storage Backend and Blockchain Connector

In order to distribute events through the Blockchain Network, the Blockchain Connector needs to be informed about changes in the Off-Chain Storage Backend. Since the Storage Backend is implemented using an [NGSI-LD](#) compliant Context Broker, this connection will happen through the “Subscription”-mechanism, as defined in the NGSI-LD specification.

To get notified about changed entities, the Blockchain Connector(s) of each access node initially subscribe to the Context Broker providing the Off-Chain Storage. The subscription will be towards a defined set of Entity-Types (as mandated by NGSI-LD 4.17.). The Entity-Types of interest are defined by the TM Forum -API implementation of the access node and should be provided to the Blockchain Connector through a standard interface.

Once the Connector subscribes successfully, the Context Broker will send all updates to the entities of interest towards the Connector. The connector now has to build the [Change-Event](#) and publish it to the Blockchain.

4.3.2.3 Blockchain Connector

The Blockchain Connector is an important component which can be instantiated alone or as part of the Access Node. It enables organisations to connect and communicate with distinct blockchain networks. The DOME project ensures that a connector reference implementation is available, which can be used as part of the access node implementation.

4.3.2.3.1 Main Functionalities of the Blockchain Connector

1. Subscribe to New Entities in the Context Broker

The Blockchain Connector allows the subscription to new entities that are stored in the Context Broker of the Access Node. This functionality enables real-time monitoring and integration of new data entities into the blockchain network.



2. Publish Entities in the Context Broker

The Blockchain Connector can publish new entities directly into the Context Broker of the Access Node. This capability ensures seamless integration of data from various sources of the blockchain network to the off-chain system.

3. Build Blockchain Events

The Blockchain Connector assists in constructing events that are ready to be published in the pre-configured blockchain network. It provides the necessary tools and interfaces to format and structure data into events compatible with the blockchain network's specifications.

4. Publishing Events in the Blockchain Network

The Blockchain Connector publishes events into the configured blockchain node of the network.

5. Subscribing to Events from a Blockchain Network

The Blockchain Connector facilitates subscribing to events from a specific blockchain network. It allows DOME participants to receive real-time updates and notifications of events occurring within the ecosystem of blockchain networks.

6. Search for Events in the Blockchain Network

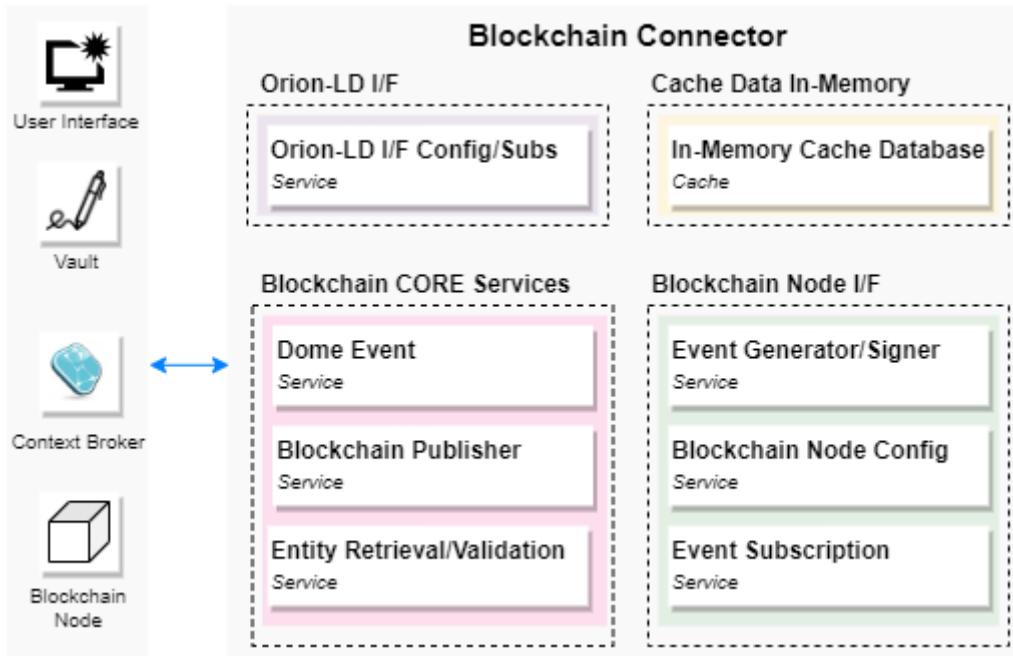
The Blockchain Connector provides the capability to search for specific events within the blockchain network. Users can query and retrieve relevant events based on defined criteria, enabling efficient data exploration and analysis within the blockchain context.

7. Retrieve Entity Data through a Blockchain Event

The Blockchain Connector facilitates the retrieval of entity data associated with a specific blockchain event. This functionality involves resolving the blockchain event and making a request to the relevant Context Broker to obtain the referenced entity.



4.3.2.3.2 APIs and Interfaces of the Blockchain connector



To implement these functionalities, the Blockchain Connector includes various APIs and interfaces. These collectively form the foundation of the Blockchain Connector, enabling seamless integration, configuration, event creation, data persistence, and interaction with external systems necessary for its functionalities.

1. Context Broker Integration Interface

This interface allows seamless integration with the Context Broker, enabling data exchange and interaction between the Blockchain Connector and the Context Broker.

2. Context Broker Connection Configuration

The Blockchain Connector provides configuration options to establish and configure the connection to the Context Broker. This includes specifying connection details such as URLs and other relevant parameters.

3. Blockchain Node Integration Interface

The Blockchain Connector offers an interface for integrating with the blockchain node. It facilitates communication and interaction between the Blockchain Connector and the blockchain network.

4. Blockchain Node Connection Configuration

Similar to the Context Broker connection configuration, the Blockchain Connector provides configuration options to establish and configure the connection to the blockchain node. Users can specify relevant parameters such as the node's URL, authentication credentials, and network-specific configurations.

5. In-Memory Cache Service

The Blockchain Connector incorporates an in-memory database that allows caching and storage of data. This system ensures efficient and reliable data management within the Connector.

6. Data Persistence System Integration Interface

This interface enables integration with the data persistence system within the Blockchain Connector. It provides methods and functionalities to interact with the persistence system, such as storing, retrieving, and managing cached data.

7. Vault System Integration Interface

The Blockchain Connector includes an interface for retrieving secrets from a Vault or other similar and secure system. This allows for signing of blockchain-related transactions or events.

8. Vault System Configuration

The Connector provides configuration options to establish and configure the connection to the Vault system.

9. Event Creation Service

The Blockchain Connector implements a service for creating events that are ready to be published in the blockchain network. This service provides methods and functionalities to format and structure data into event objects compatible with the blockchain network's requirements.

10. Event Verification Service

The Blockchain Connector includes an Event Verification API that allows users to verify the authenticity and integrity of events published in the blockchain network. This API provides methods and functionalities to validate event signatures, check event data consistency, and ensure that events have not been tampered with.

11. Entity Request Service via Event Resolution

The Blockchain Connector provides an API for creating requests for entities by resolving events. This API allows users to retrieve relevant entities based on the events published in the blockchain network.

12. Policy Retrieval Point Integration Interface

This function is just implemented by configuration for Interchain Event Distributor. The instances of the Blockchain Connector of the IED component need to implement this feature.



The interface enables integration with the Policy Retrieval Point (PRP), which allows querying distribution policies for blockchain events. It provides functionalities to retrieve and enforce policies related to event distribution through the blockchain networks.

4.3.2.3.3 Considerations about Blockchain Connector

It's important to note that every Access Node is bound to a specific blockchain network. Different blockchain connectors instances cannot coexist within a single Access Node. This ensures that the Access Node and Blockchain Connector, by extension, is tailored to the specific needs of the blockchain network, and it can function effectively to facilitate communication and delivery of event logs between the different stakeholders involved.

Only the IED component could instantiate more than one Blockchain Connector, but it is because the IED has a different nature from an Access Node. Both use Blockchain Connectors but for a different purpose.

4.3.2.3.4 Creating blockchain Events

The Blockchain Connector is responsible for generating events that will be published on the blockchain network. An event is a signed JSON in JAdES format with specific attributes added to it. These events do not contain any associated sensitive information but serve as packages of information that reference the data sources. The details regarding the event format and its claims are further explained later in the document.

4.3.2.3.5 Publishing of Event Logs

The publishing of event logs in a blockchain network involves emitting an event from a Smart Contract that has been deployed on the blockchain. Events are essentially messages that indicate that something has happened within the Smart Contract, such as a change in state or an action taken by a DOME participant.

To issue an event, the Smart Contract developer defines the event within the contract code and then calls it within a function using the "emit" keyword.

When an event has been added to the blockchain, any interested DOME participant, like a Marketplace, can access to the event data by querying the blockchain for logs that match with specific criteria. A DOME participant can subscribe to these logs, and blockchain nodes can automatically receive notifications when new events are added to the blockchain.

4.3.2.3.6 Subscribing to Event Logs

Subscribing to Event Logs in a Blockchain network is a process that enables interested parties to receive notifications about specific events that occur within a blockchain network.

All participants can set their own subscriptions using filters, which are criteria used to identify specific events of interest in the network. To enable the subscription to specific types of blockchain events, the Connector utilises the list of topics/categories used in the DOME ecosystem. This list serves as a reference for identifying the available event types that participants can subscribe to.



By allowing participants to set their own subscriptions using filters, the Connector component provides a high degree of flexibility and customization. This allows organizations to tailor their subscriptions to their specific needs and objectives, ensuring that they receive only the relevant information they need to make informed decisions and take appropriate actions.

4.3.2.3.6.1 Setting Up Subscriptions

The process for setting up subscriptions involves the following steps:

1. Retrieving the List of Topics/Categories

The Connector obtains the comprehensive list of topics or categories used within the DOME ecosystem. These topics represent different event types or classifications within the blockchain network.

2. Subscription Setup

Based on the participant's selections, the Connector establishes the necessary subscriptions to the corresponding blockchain events. It sets up filters using the chosen topics or categories as criteria to identify and filter the relevant events within the network.

4.3.2.3.7 Retrieving data of the origin Storage Back-End from resolving a blockchain event
This is the most important part of data federation because it is responsible for retrieving data stored in the source Storage Back-End based on the information received in the blockchain event. However, it does more than just executing the request. During the information retrieval process, it also includes functions such as:

- Event validation.
- Requesting the resources associated with the event.
- Adding access credentials to request the resources.
- Validation of the received entity/resource.

4.3.2.4 Blockchain Abstraction Interface

This component aims to abstract the underlying functioning of the blockchain nodes of the federated blockchain networks of DOME to be able to enable transparent interaction with them from the rest of DOME components.

For that to happen this interface will have the following endpoints:

- A configuration endpoint: Users of this interface can either use a node in a federated/delegated way (through the approved providers in DOME) or create their own Blockchain node on such networks. For the latter purpose, the interface will allow the use of their own resources (private clouds, edge servers,...) or cloud infrastructure service providers.
- An endpoint to publish blockchain events synthesised from DOME events: This endpoint allows any DOME component to use a Blockchain-based publish/subscribe model in a decoupled way.



This type of event will be published from the Context-Broker in the form of a DOME event originated from a TM Forum entity.

It is necessary to define an additional layer that translates events originated in DOME to Blockchain events to make it more efficient and aligned with privacy needs.

- An endpoint to subscribe to certain topics related to blockchain events already published.

4.3.2.7 Interchain Event Distributor (IED)

The DOME Interchain Event Distributor (IED) component offers an optimal solution to achieve interoperability between various blockchain networks, enabling applications based on one network to utilise the resources of another. As blockchain technology has matured and advanced, there has been an increasing need for blockchain projects to enhance interoperability between different blockchain networks.

The only and main function of the DOME Event Distribution Component is to disseminate the written blocks from one blockchain network to another network. However, not all written blocks will be replicated, as it depends on the configuration set by DOME Governance.

The DOME IED plays a crucial role in facilitating the replication of events across various blockchain networks. It acts as a bridge between these networks, ensuring that events written in one blockchain are accurately replicated and propagated to others.

As mentioned earlier, DOME Governance assumes the responsibility of configuring the component. This entails managing the settings and parameters required for the proper functioning of the component. The configuration process involves tasks such as registering different blockchain networks and specifying which events should be distributed.

By defining these rules, DOME Governance controls the flow of information between participating blockchain networks. When distributing events using the Interchain Event Distributor, additional policies may apply, such as preventing the distribution of events to certain blockchains based on governance decisions. This implementation of IED is linked to the implementation of a Policy Retrieval Point (PRP) associated with a Policy Administration Point (PAP) to manage Distribution Policies. It's worth noting that this situation can disadvantage event consumers on certain blockchains compared to consumers on accepted blockchains, so it needs to be considered by governance as well.

Each blockchain network has a node that plays a pivotal role in the replication process. Nodes serve as the backbone of a blockchain network, responsible for maintaining a copy of the entire blockchain and participating in the consensus mechanism. Disseminating events across different blockchain networks involves the active propagation and synchronisation of events by nodes to ensure consistency and transparency throughout the federated ecosystem.

The enabling technologies to create this component are introduced in point **5.4.2 The Data Flow** and will be elaborated in the following project iterations.

4.4 Data Federation

Data Federation refers to the process of aggregating and integrating data from multiple sources, usually distributed across different systems or organisations, into a unified and



coherent view. It involves combining data from diverse sources while preserving their autonomy and original structure.

4.4.1 The model of transferring data in a blockchain.

The model for sharing data between organisations is the **Pull Model**, instead of a push one. With the pull model the data remains in control of the source entity, which can provide granular access to authorised participants when they require that data.

In our use-case, any participant (origin) is authorising other participants (target) to access a subset of the data held on-custody in the origin participant. This authorization could be very simple or arbitrarily complex, for example with time limits, for a specific purpose, etc.

This model can be considered when some (or all) of the following requirement have to be met:

1. **Data on-custody:** the data is managed by a legal entity, not by a natural person. If the data belongs to another entity, especially for a natural person, it is managed using an on-custody model. Depending on the specific circumstances, that entity may be a Data Controller or a Data Processor, but in any case, it is subject to very strict legal requirements as set forth by GDPR.
2. **Strong privacy:** it may not be desirable to disseminate the data among all the nodes (or even a subset) in the network in a proactive (or push) model, even if encrypted.
3. **Pull (on-demand) access:** the data can be accessed by the target organisation only when needed for processing, not necessarily at the same moment when the data is available.
4. **Large amounts of data:** the data to be shared with other participating nodes is large (documents, PDFs, videos, audio, etc.).
5. **Auditability of access:** we need to know who has accessed what data, when and for what purpose. This is better done in a pull model than in a push one.

4.4.2 The Data Flow

In the context of a Service Provider and a Marketplace, we will explore an example that demonstrates the flow of federated data between these two entities. The Service Provider acts as a data source or provider, while the Marketplace serves as a platform where various services and data are offered.

To enable data federation between the Service Provider and the Marketplace, a federated data integration approach is utilised. This approach involves establishing a connection or interface that allows the Service Provider to selectively share specific data with the Marketplace while retaining control over its data assets.

To not extend the description of all data flows between the different participants of DOME, we would like to explain a baseline case that more or less fits with all the possible interactions between participants of the ecosystem. In the following data flow, both participants are



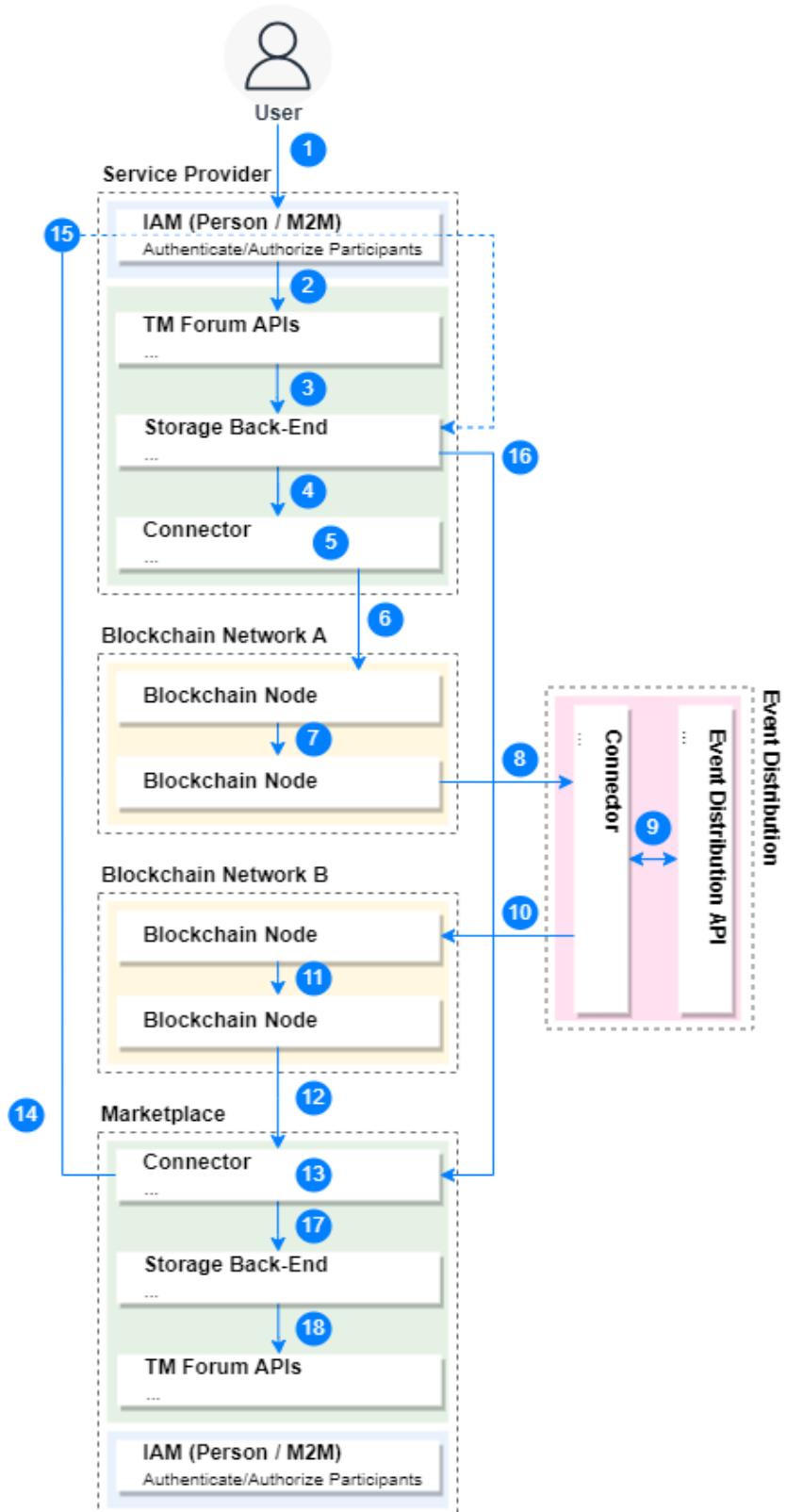
connected to distinct blockchain networks, we chose this flow to exemplify how the Event Distributor component interacts in these types of transactions.

1. A participant who is registered with the Service Provider accesses their portal to create a new entity, such as a Product Specification or Product Offering.
2. If the participant's authentication and authorization are successful, they are granted permission to create the new entity using the Service Provider TM Forum APIs.
3. The Service Provider TM Forum API saves the newly created entity in the off-chain Service Provider Storage Back-End, which is represented by a Context Broker instance.
4. The Service Provider Storage Back-End notifies all subscribed applications/components about the storage of the new entity.
5. The Service Provider Connector, which is subscribed to the Service Provider Storage Back-End, receives the resource's URI and retrieves the entity. Upon receiving the new entity, the connector performs data validations and checks for publishing policies.
6. If all the validations are successful, the Service Provider Connector generates a blockchain event and publishes it to the blockchain node using a specific interface.
7. A blockchain transaction is created, containing a secure pointer (hash) that refers to the data (the event does not contain the data). This pointer is stored on-chain and will be shared with a subset of the entire blockchain network, which represents the target organisation(s).
8. When a new block arrives at the distribution node, which is a blockchain node configured for blockchain federation, it triggers a notification to the Event Distribution.
9. The Event Distribution Connector receives the event and performs various checks, including validating the event's signature, issuer's validity, timestamps, and more. If all the checks pass, the Event Distribution Connector requests the Distribution Policies through the Event Distribution API. If there are no conflicts, the Event Distribution signs the new transaction without modifying the event's data.
10. Each Event Distribution Connector distributes the event to its respective configured blockchain network, only if the event has not been previously registered in the target blockchain. Events have globally unique identifiers, essential for deduplication in case of double registration.
11. The same process as described in point 7 is repeated.
12. The Marketplace is subscribed to events on the Blockchain Node and applies filters to select events of interest. When a subscribed event is triggered, the Marketplace Connector examines the corresponding transaction within the blockchain and retrieves the event.
13. The Marketplace Connector applies various checks to validate the event's signature, issuer's validity, timestamps, and other factors. If the event passes these checks, the connector initiates the Entity Retrieval Flow.



14. The Marketplace Connector uses the data from the event to request access to the Entity from the origin organisation's store, which is the Service Provider Storage Back-End.
15. The request is intercepted by the Service Provider IAM (Identity and Access Management) system, which grants access to the data only if the target organisation is authorised. In the case of personal data managed by the origin organisation (acting as the data controller or data processor under GDPR), the origin organisation verifies that the target organisation (Marketplace) has explicit consent from the owner of the data for this action. An SSI (Self-Sovereign Identity) framework may be employed to facilitate this process, although further details are to be discussed later.
16. The Service Provider Storage Back-End retrieves and serves the requested entity as a response to the received request.
17. The Marketplace Connector validates the received entity and stores it in the Marketplace Storage Back-End.
18. The Marketplace Storage Back-End notifies the TM Forum APIs that a new entity has been stored.





4.5 Data considerations

4.5.1 The Blockchain Event

The blockchain event is the fundamental object used in the Shared Catalog and Transaction Ledger to facilitate transactions and track changes within a blockchain network. It consists of a JSON object with several defined attributes that capture essential information about the event.

1. Event-type

The Event-type attribute indicates the type or category of the event. It represents the nature or purpose of the event within the blockchain network, allowing participants to distinguish between different event types and handle them accordingly.

2. Timestamp

The Timestamp captures the exact time when the event is built. It serves as a chronological reference, enabling participants to establish the temporal order of events within the blockchain network. The timestamp ensures data consistency and provides a reliable record of when the event occurred.

Please note that, at the time of writing the DOME AR document, the JAdES signature format does not support Qualified Timestamp (Baseline-T). As a result, the decision has been made to utilise a Non-Qualified Timestamp (Baseline-B) as an alternative solution. This temporary measure allows for the continued implementation of timestamp functionality until a suitable solution for incorporating Qualified Timestamps can be found.

3. Data location

The Data location specifies the storage or location of the data associated with the event. It denotes the address or identifier of the data within the blockchain network, facilitating efficient retrieval and verification of the event data. To expand on this point in the section, Safe pointer to the origin data in the blockchain transaction: the Hashlink of the document.

4. Filter relevant meta-data

The Filter relevant meta-data contains additional information or metadata related to the event. This metadata provides context and relevant details about the event, which can be used for filtering or processing purposes. Participants can utilise this attribute to extract specific information relevant to their needs, or apply filters to manage and handle events effectively. To expand on this point in the section, Type of the Entity: the Metadata.



4.5.1.1 Safe pointer to the origin data in the blockchain transaction: the Hashlink.

There are different approaches to creating a secure pointer, but one simple way is to include a Uniform Resource Identifier (URI) and a hash of the original data in a transaction. This helps to ensure that the data hasn't been tampered with after the transaction is added to the blockchain while also enabling the requester to identify outdated data.

One commonly used method to achieve this is through the [IETF Hashlink System](#). Once the Hashlink is stored in the blockchain, other organisations can confidently assume that the data retrieved from the original organisation is unaltered, provided that the hash of the data matches the one in the Hashlink. This makes it possible to safely and efficiently transfer data between organisations using the power of the blockchain, while maintaining privacy and security. It allows the marketplace to validate consistency, and auditing.



4.5.1.2 Type of the Entity: the Metadata

The type of the entity supports filtering on the Marketplace side and potential to batch multiple events on the Service Provider side – further reducing transaction costs.

4.5.2 Security considerations about Events or Logs in Blockchain transactions.

Events, in blockchain transactions, also need to be considered when it comes to security. Security considerations for events in blockchain transactions revolve around ensuring data integrity, preventing information leakage, auditing the transactions, and distributing policies to relevant parties.

4.5.2.1 Data Integrity

Data integrity is a crucial aspect of security, ensuring that the data in a transaction is not tampered with or modified in any way.

Every transaction in the Blockchain is signed by its origin, and the Event source can be verified.



The Events contain a hash of the corresponding entity, ensuring that the integrity of the entity can be verified by the target participant after retrieval from the origin.

4.5.2.2 Information Leakage

Information leakage, on the other hand, refers to the possibility of sensitive information being revealed to unauthorised parties during an event in a blockchain transaction.

Events do not contain the actual data, which limits the visible information through the Blockchain. Data might still be revealed through the added meta data, which must be considered by the data provider when publishing an event.

The actual Data will be retrieved from the origin participant, and an additional Security Framework can be applied, if necessary.

4.5.2.3 Auditing

Auditing helps to track and monitor transactions, ensuring that any suspicious activity is detected and prevented. Blockchain, as an immutable store of Entity-Hashes, allows auditing the Entity-History.

4.5.2.4 Policy Distribution

Policy distribution ensures that all relevant parties are aware of the security policies in place for events in blockchain transactions. Policies can be viewed as a type of data that can be distributed using similar mechanisms as other forms of data. This means that policies can be transmitted through the same channels as other information, allowing for efficient distribution and dissemination.

When it comes to the implementation of policies on the blockchain, events corresponding to these policies are typically published to the blockchain. Meanwhile, the specific policy itself is retrieved from the Storage Back-End. This ensures that policies can be easily managed and updated, while still being accessible to those who need them.

Finally, policies can be applied on the origin/target Access Control Layer. This layer is responsible for ensuring that only authorised individuals or entities are granted access to specific resources or functionalities within the marketplace ecosystem. By applying policies on this layer, organisations can ensure that their solutions remain secure and operate in accordance with their specific needs and requirements.

-



4.6 Implementation Considerations

4.6.1 Joining the Marketplace

This section delves into implementation considerations concerning data within the DOME ecosystem. When implementing a new Marketplace in the ecosystem, establishing the initial connection with the Distributed Persistence Layer becomes of paramount importance. This connection enables the new Marketplace to retrieve all the essential data required to operate autonomously within the ecosystem. The initial data dump is facilitated through the configuration of the Access Node, ensuring a seamless transfer of the necessary information.

Note:

In the current architecture the Access Node is bound to one and only one Participant (ex. service provider, marketplace) and multiple Participants cannot use the same Access Node.

Binding an Access Node to a Participant means that a separate deployment of the Access Node is required for each Participant.

Consequences of this model are:

1. It raises the entry barrier in the DOME ecosystem for Participants, by conditioning participation in the federation on hosting an Access Node upfront.
2. It denies Participants the ability to reduce operating costs when they could otherwise share the same Access Node in a trusted infrastructure.

It is planned that in a future iteration the Access Node will be implemented so as to be capable of hosting multiple Participants in isolation from each-other (i.e. make the Access Node multi-tenant capable). It will allow full API-driven configuration of the Access Node (i.e. via an Admin API)

The benefits of this enhancement are:

1. Lowers the entry barrier by allowing a Participant to onboard onto an already existing Access Node whose hoster it trusts, while still being able to migrate to a self-hosted Access Node at a later time.
2. Allows Participants to save on hosting and operating costs by sharing a single Access Node as tenants.

As a concrete example in DOME, Dawex' Data Exchange Platform technology allows customers to deploy and run their own Marketplaces as software-level tenants on a shared infrastructure, thus optimising hosting costs. In contrast to that, under the current constraints, the Dawex platform would have to create a full Access Node deployment for each tenant (as each tenant would be a distinct Participant in DOME). If the Access Node were capable of multi-tenant, the platform would need a single Access Node deployment to securely cater for all Participants.

However, it is crucial to emphasise that this initial data transfer must align with stringent security policies. As data is being transferred from the Distributed Persistence Layer to the



new Marketplace, robust security measures must be in place to safeguard the integrity and confidentiality of the data.

Furthermore, the initial configuration of the Access Node should be carefully planned and executed. This includes defining data retrieval parameters, setting up appropriate filters, and configuring data synchronisation processes. The goal is to establish a seamless and efficient data transfer mechanism that empowers the new Marketplace to operate with complete autonomy and access the necessary information for its functionalities.

To do that, we consider that a good approximation could be:

- The blockchain node contains the complete (immutable) event history.
- Connector processes the complete history, retrieving the last event for each entity.
- Connector retrieves and stores the entities from the Service Provider.



5 DOME marketplace features

On top of the foundation technologies introduced in the previous chapters, the DOME project will realise an open standard-based, vendor-neutral reference framework to support the materialisation of the envisioned federation of marketplaces.

Among those marketplaces, a notable instance is represented by the DOME marketplace itself, operated according to provider-neutral, inclusion and equality principles. The DOME central marketplace will provide the needed backend and portal services for a) service providers to administer the description of their services and monitor their lifecycle as well as for b) end users to discover services and transition to marketplaces through which they will activate them for actual use, typically on environments setup on the cloud or edge by IaaS or Platform providers.

This chapter provides the scope for the features that the DOME marketplace will deliver to involved stakeholders, initial design of the processes involved in a federated procurement scenario and high-level architecture of the various subsystems delivering such features.

In particular, the first part of the chapter will introduce core concepts, terminology and an overview of the procurement process. Each activity in this process will be explored individually, following the whole journey from user subscription, through catalogue management, ordering, provisioning, tracking, billing and invoicing up to payment.

The second part will focus on added value services behind the DOME central marketplace, namely NLP-enabled search capabilities, advanced reporting and analytics features, and user support capabilities powered by an AI-based chatbot.

5.1 Background

5.1.1 Concepts and Terminology

For the sake of describing behaviours within the DOME ecosystem and structure of the various DOME services, it is worth setting some background concepts and terminology that will be used throughout this section:

DOME Ecosystem is the technical and business environment where procurement activities happen, following established rules and complying with technical specifications. Activities (catalogue management, ordering, billing, tracking, etc.) are recorded and potentially visible to the whole federation according to identity management and access control mechanisms.

Federated Marketplaces are B2B and B2C platforms hosting product listings from one or multiple sellers, facilitating them to meet customers and conduct business with them. A marketplace, as opposed to a catalogue, also exposes some kind of online transaction capabilities. A marketplace listing products from one single seller (who, usually, also governs the platform) is commonly referred to as an **eCommerce** platform.



DOME Marketplace is actually a federated marketplace, run by a DOME Operator (that will be established within the project). It will be operated under certain governance rules avoiding a dominant position in the federation and ensuring sustainability. However, from a technical perspective, it behaves as a federated marketplace.

DOME Services (the ones described in this chapter) are primarily exposed through the *Portal* of the DOME marketplace but can also be exploited by federated marketplaces and providers via APIs, to enhance/complement the features of their marketplaces.

Service and Resource Providers are vendors of cloud solutions being offered in the ecosystem. Providers can be:

- linked to a federated marketplace with legacy integration with them in terms of various processes (catalogue, ordering, provisioning, etc..)
- linked to a federated marketplace with some degree of compliance with DOME specifications, thus enabling a deeper direct involvement into DOME processes (e.g. provisioning, tracking, reporting, ...) otherwise only possible through the linked marketplace.
- linked to the DOME marketplace, thus complying (to different extents) to the DOME technical specifications. Interaction between the DOME marketplace and the provider only happens via DOME specifications (i.e. TM Forum APIs); no legacy integration can happen here.

Resellers are organisations active on one or more marketplaces (either DOME or federated) selling services and resources from other providers, sometimes in a bundled offering.

Consumers are individuals, business consumers or public entities procuring services and resources in the DOME ecosystem.

Marketplaces Operators are business entities that run a marketplace with any selection of possible features (e.g. catalogue, ordering, billing, payment, etc.). We can identify different possible combinations with other roles (although this does not change the role and responsibility of being a marketplace operator):

- independent marketplaces hosting services and resources from third-party providers.
- marketplaces ran by individual providers, typically selling their own products (e.g. Commerce platforms)
- the DOME operator, running the DOME Marketplace exposing offerings from providers/marketplaces in the federation

As introduced earlier, the same entity can play different roles in the ecosystem (e.g. a provider can also run a own marketplace, business consumers can also be service providers, integrators can bundle own services with others and publish them in the DOME catalogue acting as a provider and reseller, etc.)

Finally, DOME will provide means for integration of third-party services:

Certification and Audit Agencies which will help to validate the reliability, security, and sovereignty of certain cloud services by checking/verifying their compliance with predetermined market-wide certifications.

IAM service providers offering services aligned with open standards for IAM adopted in DOME, bringing participants the ability to securely manage identities and access to specific cloud and edge data/app services.



Billing and Payment service providers working as gateways that rely on transaction logs registered in the DOME decentralised persistence layer to provide secure, transparent and trustful billing to consumers and payment to providers.

5.1.2 The procurement process

The procurement process refers to the series of steps involved in delivering and provisioning cloud services to customers. It encompasses the entire lifecycle, from the initial request for a service to its successful deployment and activation. Here's an overview of the fulfilment process in cloud environments in its general form (i.e. without entering the details of a federated scenario):

Catalogue Management: the cloud service provider maintains a service catalogue that lists all available services along with their descriptions, pricing, and service-level agreements (SLAs). The service catalogue serves as a reference for customers to choose the appropriate service based on their requirements.

Order Placement: the procurement process begins when a customer or end-user submits a service request. This request typically includes details such as the desired cloud service, specifications, configuration requirements, and any additional preferences or customizations.

Order Management: once the customer submits a service request, the service order management system processes and validates the request. This involves verifying the customer's identity, checking the availability of the requested service, and validating any dependencies or prerequisites.

The processing of an order also includes a series of actions that the provider needs to take in order to deliver the requested service to the consumer. Such activities might include:

Provisioning and Configuration: allocating the necessary computing resources, networking components, storage, and any additional software or tools required to set up the requested service. The provisioning process can vary depending on the type of service (e.g. Infrastructure as a Service, Platform as a Service, Software as a Service).

Deployment and Activation: once the provisioning and configuration are completed, the cloud service is deployed and activated. This involves setting up the necessary infrastructure, configuring network connectivity, installing software components, and performing any required configurations or customizations based on the customer's specifications.

Testing and Quality Assurance: after the service is deployed and activated, thorough testing and quality assurance procedures are conducted. This ensures that the service functions correctly, meets performance benchmarks, and adheres to specified SLAs. It involves conducting functional testing, load testing, security testing, and any other necessary validation processes.

Service Delivery: once the service passes all the required tests and quality checks, it is ready for delivery to the customer. The cloud service provider notifies the customer about the successful deployment and provides him with the necessary access credentials, documentation, and support materials.

It is worth noticing that the four above sub-processes entirely happen under the control and visibility of the service provider, while DOME takes no part in their orchestration. Whenever agreed by the service provider, DOME (referring to backend services and, thus, marketplace users) might be informed about the status and progress of the procurement process.



As the service is delivered to the consumer, other relevant processes are activated:

Service Monitoring and Management: after the service is delivered, the cloud/edge service provider continuously monitors and manages the service to ensure its ongoing performance, availability and security. This includes monitoring resource utilisation, addressing any service disruptions or incidents, and applying updates or patches as necessary.

Usage Tracking: the process of collecting usage information, their processing, aggregation and structuring, and their publication to interested parties for the goal of charging the customer for their cloud service usage.

Billing and Invoicing: the procurement process also involves the billing and invoicing activities. The service provider generates invoices based on the agreed-upon pricing model (e.g. pay-as-you-go, subscription-based) and sends them to the customer.

Payment: the process of transferring due amounts for purchased products from customers to federated marketplaces and/or service providers, including one-off (interactive) payments as well as recurring (non-interactive) payments.

The diagram below shows a simplified view of major interactions among DOME federated entities (i.e. marketplaces, providers and consumers). The details of each of them will be explored in the rest of this chapter.

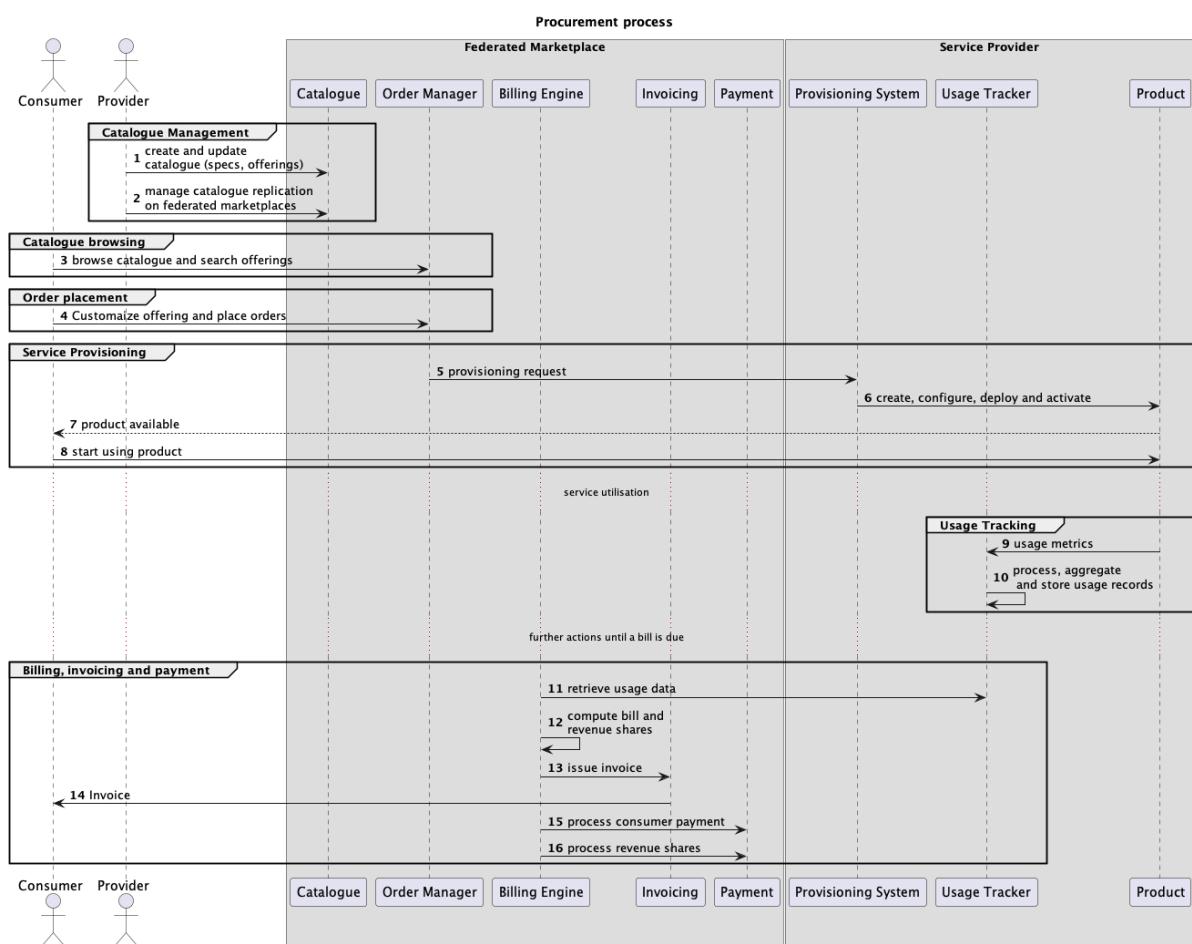


Figure 40 - The procurement process



5.2 Subscription Management

The On-boarding process allows new participants to join DOME. Organisations can register themselves and establish the roles they wish to play by using the DOME Trust and IAM framework. This process creates an account for the organisation that firstly needs to be verified throughout a qualification process, where any necessary information and documentation needed to validate the participant is asked to be provided. In addition, the participant is required to accept the DOME Terms and Conditions and sign the contract establishing the Revenue Sharing and the Code of Conduct. The detailed description of the on-boarding process is defined in section 3.

Once the new participant is verified, it will be recognized as a valid issuer. This means it will be authorised to issue Verifiable Credentials to their organisation members covering the roles the participant is playing. From this moment on, they can access the DOME Marketplace.

It is also possible for participants to terminate their accounts at any moment as long as there are no pending payments, ongoing complaints and/or active contracts. In this case, all the account and personal information is removed from the DOME systems.

From the point of view of the DOME Marketplace, all participant information is managed using the TM Forum APIs and thus stored as part of the DOME Persistent Layer. In particular the following APIs are used to handle participants:

- **Party Management API:** Provides a standardised mechanism for the management of parties including both Individuals and Organizations. Such parties are referenced from all TM Forum models in order to detail which parties are involved in the different TM Forum entities.
- **Party Role Management API:** Provides the API needed to manage the roles a particular party is playing, including roles of the platform itself and roles coming from an established agreement with other parties.
- **Account Management API:** Provides a standardised mechanism for the management of party accounts, including Billing, Settlement and Financial accounts.

Those entities included in the TM Forum APIs will be linked with the participant information handled by the Trust and IAM Framework (Wallet and VC). In this regard, during the first access to DOME made by an organisation LEAR, the system will automatically create the Organization profile in the Party Management API and the initial set of platform roles in the Party Role Management API using the information included in the credentials. Then, the participants will be able to update their profiles and create their accounts (Billing, Settlement, etc.) using the DOME Portal or directly through the TM Forum APIs using the proper Verifiable Credentials.

5.3 Catalogue Management

Both DOME and Federated Marketplaces will allow service providers to publish their catalogues of services, so that they can be discovered, browsed and acquired by potential customers. To support such a feature, DOME will rely on the TMForum APIs for registering the different service and resource specifications, create product offerings and categorise such offers so they can be easily filtered.



This section describes how the different specifications, product offerings and categories are managed in DOME as well as how catalogues can be browsed. In addition, it is described how the different catalogues will be shared across the federated environment and how its visibility will be established.

5.3.1 Managing Specifications for Products, Services and Resources

The different Products that will be available and offered in the DOME Marketplace are made up of a set of Services that can be used by the customers (for example, a computing service), and a set of resources consumed or used by such services to work. Those Resources are physical or non-physical components (or some combination of these) within an enterprise's infrastructure or inventory (for example a physical port assigned to a service).

In order to monetize products within the DOME Marketplace, including the provisioning of the acquired services and the allocation of all the needed resources, DOME Marketplace needs to manage the characteristics and the information of such products, services and resources. To do so, DOME Marketplace will use the TM Forum APIs. In particular, the specification models included in the Product Catalog Management API, Resource Catalog Management API and the Service Catalog Management API.

The specifications managed in DOME are:

- **Resource Specification:** It provides the generic means for implementing a particular type of Resource. In essence, a Resource Specification defines the common attributes and relationships of a set of related Resources.
- **Service Specification:** It offers characteristics to describe a type of service. Functionally, it acts as a template by which Services may be instantiated. By sharing the same specification, these services would therefore share the same set of characteristics.
- **Product Specification:** It is a detailed description of some of the attributes that will characterise Products created when a Product Offering is successfully ordered.

The following diagram depicts how a Product Specification is registered in DOME, including the Resource Specification and Service Specification that made it.



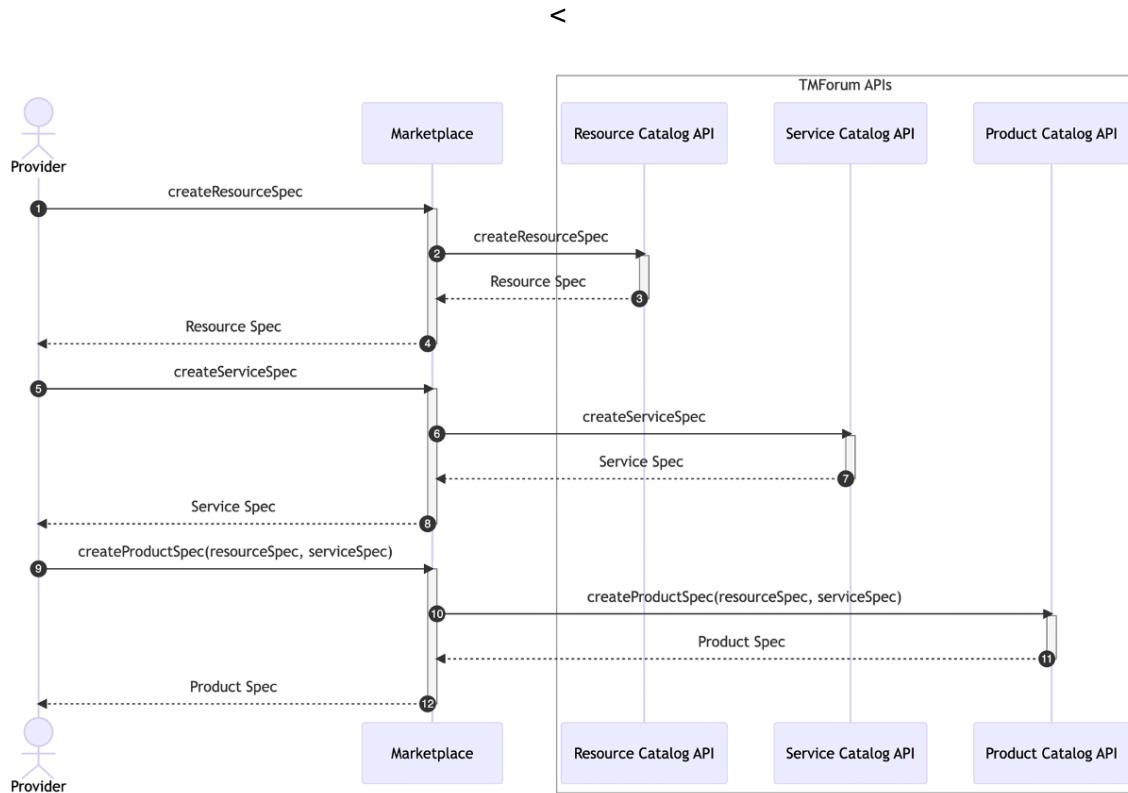


Figure 41 - Managing resource, service and product specifications

5.3.2 Managing Categories

Categories are used to group Service and Resource Candidates, as well as Product Offerings into logical containers. Categories can be organised in a tree-like structure to ease the browsing and selection process through filtering.

To enable effective filtering and comparison on the catalogue, the set of categories must be centrally managed. In particular, the DOME Marketplace will act as the reference source for the tree of categories, and the DOME Operator, following governance rules, will be in charge of its management.

Within DOME, the management of categories will be grounded on concepts, recommendations and specifications from TMF620 "Product Catalog Management API".

5.3.2.1 Categories Lifecycle States

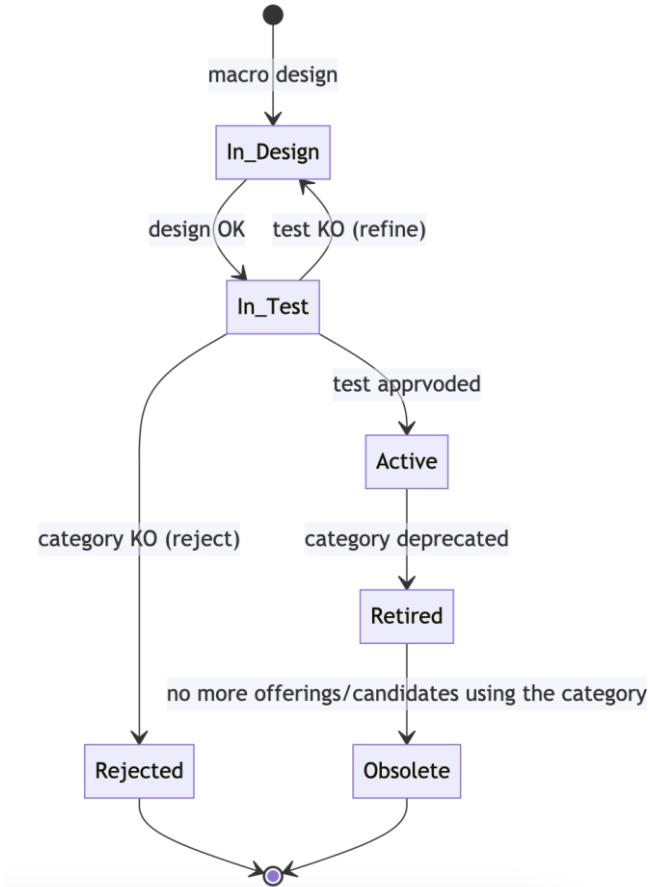


Figure 42 - Lifecycle of categories

Categories used to group offerings and candidates are introduced in the DOME ecosystem

following a specific lifecycle, whose states are a subset of those recommended by TM Forum APIs for all managed entities.

When the conception of a category element is started, its first status is “**In Design**”. The purpose of this status is to start tracking the category into the platform and allow its refinement in terms of naming and description.

As the definition is mature enough, the category is moved to the “**In Test**”, meaning that it can be attached to some pilot candidates and offerings. Marketplace functions (e.g. search, browse) leveraging these categories should ensure explicit user awareness, both when activating the function and when presenting results.

Any failure (e.g. negative feedback, poor adoption, duplication, etc.) in testing the adoption of a category, might require a further refinement (back to ‘Design’ status) or lead to a **Rejection**, with no further action.

Conversely, a positive feedback in the test stage might lead to the '**Activation**' of the category, meaning that it's ready for wider and unconditioned usage within the ecosystem.

Categories that are planned to be removed will enter the '**Retired**' state. This means that this category cannot be attached to any new offering/candidate but can still remain attached to older ones.

Finally, when a '**Retired**' category is no longer held by any offering/candidate, it can be made '**Obsolete**', meaning that it can be safely removed from the environment.

5.3.2.2 Creating categories

Categories are created, typically within the DOME central marketplace, through the "Create Category" operation specified within Product Catalogue Management API.

Upon creation of a new category, a "Category Create Event" is issued and disseminated to the whole federation through Decentralised Persistent Layer mechanisms; interested parties are then able to retrieve the Category Resource from the DOME marketplace. Dissemination and retrieval processes are executed as described in section 4.4.2 The Data Flow.

Note that although every federation member is, in principle, allowed to issue a "Category Create Event" and expose "Category Resources", only those exposed by the DOME Marketplace can be used to enrich product offerings, offering and service/resource candidates. Compliance with the DOME category tree will be enforced during catalogue management processes.

5.3.2.3 Retrieving categories

Categories can be retrieved, both from local storage and from federated marketplaces (most likely only the DOME central marketplace, in this case) through operations enabling bulk ("List Categories") and individual ("Retrieve Category") retrieval, featuring filtering and attribute selection.

5.3.2.4 Updating categories

Categories are updated, typically within the DOME central marketplace, through the "Patch Category" operation specified within Product Catalogue Management API. Categories can be updated in terms of all their attributes and sub-categories. Whenever this happens, the change is spread in the federation by issuing a "Category State Change Event", following the same process used for the creation.

Upon update of a category, federated marketplaces / providers might need to update offerings and candidates in their catalogues accordingly (e.g. when a category is 'retired'). This, in turn, will trigger catalogue synchronisation processes in the ecosystem.

Catalogue consumers (e.g. other federated marketplaces) should be tolerant when processing offerings and candidates that refer to categories whose lifecycle state is not active. In such a case, the reference to the non-active category should be simply ignored.

5.3.2.5 Deleting categories

Categories are deleted, typically within the DOME central marketplace, through the "Delete Category" operation specified within Product Catalogue Management API. Whenever a



category is deleted, the event is spread in the federation, by issuing a ‘Category Delete Event’, following the same process used for the creation and update.

Upon deletion of a category, federated marketplaces / providers should update offerings and candidates in their catalogues accordingly. This, in turn, will trigger catalogue synchronisation processes in the ecosystem.

Catalogue consumers (e.g. other federated marketplaces) should be tolerant when processing offerings and candidates that refer to deleted categories. In such a case, the reference to the missing category should be simply ignored.

5.3.3 Managing Offerings

DOME Marketplace will allow customers to acquire the different products and services published by service providers. To support such a feature, offer information will be handled using the Product Offering model as defined by the TM Forum Catalog Management API. A Product Offering represents an offer on products that are orderable from the different providers. It includes pricing and agreement information, the reference to the Product Specification, and other characteristics of the Product that gets created when a Product Offering is procured. In addition, it includes the references to the Resource Candidates and Service Candidates of the Service and Resource Specifications that made up the Product Specification to support the instantiation in the different inventories.

5.3.3.1 Offerings lifecycle

All the different models defined as part of the TM Forum Product Catalog Management API share the same lifecycle, including the Product Offering. The following picture depicts the lifecycle of a Product Offering:



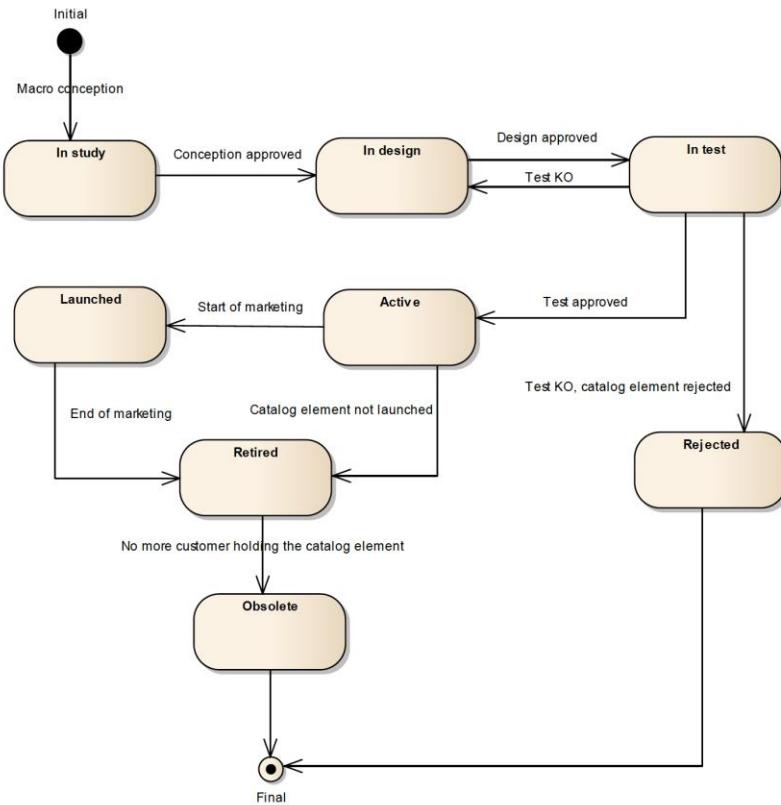


Figure 43 - Lifecycle of offerings

It can be seen that the Product Offering lifecycle includes statuses not only for the commercialization phase, but also for its conception and design:

- **In study:** The concept for the new product offering is being analysed
- **In design:** The concept has been approved and the product offering is being designed
- **In Test:** The offering design is ready and it is being tested
- **Active:** The design is approved and the tests are OK, the product offering is ready
- **Rejected:** The design is not approved and the tests are not ok, the product offering is rejected
- **Launched:** The product offering is available to be acquired by customers
- **Retired:** The product offering is no longer available to be acquired by customers, but customers who has already acquired it still has support
- **Obsolete:** The product offering is no longer maintained by the service provider

It is important to remark that the initial statuses of the lifecycle are intended for scenarios where the development of the Product Offering is relevant. For the particular case of the DOME Central Marketplace it is assumed that the services and resources offered are already available, so the lifecycle used will start on the Active status.

5.3.3.2 Offerings management in the DOME central marketplace



The DOME Marketplace will implement the whole TM Forum Product Catalog API, allowing providers to create, update and delete Product Offering models.

Service Providers will be authorised to create new Product Offerings in the DOME Marketplace in Active state, including the product specification that is being offered and the product catalogue where the offering is being made available. Such specification and catalogue must exist in advance before the creation of the Product Offering.

While the product offering is in Active state, it is not available to be discovered and cannot be placed into a product order, so the service provider will be able to update any information, including descriptions, pricing models, categories, and so on before its commercialization. Once the status of the product offering is set as Launched, such an offering is made available in the search and browsing section of the DOME Marketplace and can be discovered and acquired by the potential customers. As the product offering references product specifications and product catalogues, both elements must also be in the Launched state before launching the product offering.

The service provider will be able to update the information of the product offering at any moment while it is in Active or Launched state. Note that even if the Product Offering has been already acquired by some customers, the conditions, including characteristics, pricing model, etc. are copied to the Product Order and then placed in the Product Inventory as described in the section 5.4.2 “Order Placement”, so updating product offering information of a Launched offering does not affect orders already placed.

Finally, the service provider will not be authorised to delete product offerings directly from the DOME Marketplace. To stop the commercialisation of an offering, its status must be set to Retired, so customers who have already acquired the product offering can keep using it (according to the established agreement and the offering terms and conditions), and at the same time, it cannot be acquired by new users. DOME Marketplace admins will be able to remove Product Offerings from the system.

5.3.3.3 Agreements management

Registering product offerings in the DOME Marketplace allows potential customers to acquire those different services offered in it. This enables the basic customer-provider scenario where customers can place orders (as defined in section 5.4.2 “Order Placement) and pay accordingly to the selected pricing model. Nevertheless, this scenario is not the only one that can be supported by the Marketplace using the TM Forum APIs. The Agreement Management API can be used in order to support any agreement between parties, not only purchases but also any contract or arrangement such as a service level agreement or a collaboration. Such agreements may involve different products, services and resources and can include multiple parties involved, playing different roles.

The DOME Marketplace will allow parties to register potential agreements by supporting the Agreement Specification model defined by the TM Forum Agreement API. An agreement specification defines a template that can be used when a new partnership is established, and can include any number of agreement characteristics defining the conditions of the potential agreement. These templates can be used for establishing agreements as defined in section 5.4.2.2.3.

5.3.3.4 Revenue sharing policy definition

The DOME ecosystem incorporates multiple organisations playing different roles aimed at developing business. In some scenarios, different services can be bundled together to



generate added value or can be resold if the terms and conditions allow such a business model. To deal with these use cases, section 5.4.6 “Revenue sharing” defines the approach taken in DOME for the reselling process and the required revenue sharing.

From a technical perspective, the DOME Marketplace will allow service providers to manage their revenue sharing policies and the rules included in them, enabling providers to register, update and remove such policies.

From the point of view of product catalogue management, when a new product offering reselling a service is being registered, the DOME Marketplace will allow attaching a revenue sharing policy previously defined. With such an approach the DOME Marketplace will be able to apply the revenue sharing policy to the incomes generated by a particular product offering and distribute the revenues accordingly.

5.3.4 Browsing the DOME central marketplace catalogue

The DOME Marketplace can potentially include a huge number of product offerings. In this regard, it is critical to provide proper browsing and searching capabilities in order to increase the usability, allowing potential customers to discover the offers that best fit their needs.

To do so, on the one hand, the DOME Marketplace will support categories as defined in section 5.3.2 “Managing categories”, enabling categorisation of the published product offerings. Potential customers will then be able to filter and easily discover those offerings. Categories will be created and managed by the owner and administrators of the marketplace.

On the other hand, the DOME Marketplace supports the creation of Product Catalogues, as defined by the TM Forum Product Catalog API. These product catalogues are created by the different providers and allow them to publish their offerings in such catalogues, so they can be classified according to the providers' needs.

The DOME Marketplace will allow customers to select the different published catalogues so they can browse the product offerings included in them. In addition, as the different providers can create multiple catalogues, the DOME Marketplace will allow users to search the catalogues themselves using the description information given by the catalogue provider.

5.3.5 Replication and visibility of catalogues in the federation

For marketplaces and service providers, one of the main benefits in joining the DOME federation is the potential it delivers in increased visibility of their cloud and edge offerings to all customers across Europe.

In fact, DOME enables federated marketplaces operators and service providers to expose selected parts of their catalogues through other market channels. This will also reduce their administrative burden as typically they would have to adhere to the rules of specific marketplaces, typically run by individual cloud IaaS/platform providers.

The low barrier for becoming visible through the federated marketplaces, linked to the shared catalogue of cloud services, will stimulate competition among marketplace providers, which ultimately will turn into better conditions from marketplaces as intermediaries and overall growth in number of end users.



For such an approach to be acceptable by marketplaces, both parties of the relationship (the source provider/marketplace and the exposing marketplace) must have full control over the process.

5.3.5.1 Establishment of replication and visibility policies

In order to control how and where product offerings are replicated in the DOME ecosystem, marketplaces and product/service providers should have the means to set **replication policies** based on different criteria to be matched by the exposing marketplace (geographical, domain, time-based, legal, compliance, specific marketplaces, etc.).

Also, the offering owner can further address a specific customer base within the same target marketplace by establishing **visibility rules** that determine who can get access to the offering based on different criteria (i.e. customer properties, etc.).

Both replication and visibility policies generally apply to individual offerings and to entire catalogues. However, there might be cases that even some specific offering metadata can be subject to visibility constraints (e.g. the price in an offering is only visible to registered consumers and not to anonymous ones).

It's important to highlight that such policies are to be meant as a desiderata for marketplaces and service providers and do not imply any action on target marketplaces that are not willing to be involved in the replication process.

5.3.5.2 Establishment of acceptance rules on federated target marketplaces

On the other side of the catalogue replication process, the target marketplace has always full control over the content of its exposed catalogues. Whatever the replication and visibility requests are, it is free to accept or reject any offering on behalf of other marketplaces and providers.

The specific approach taken by marketplaces within the DOME federation is outside the scope of this design document. However, the DOME central marketplace will expose at least two mechanisms supporting the replication process:

1. the capability to set acceptance policies (based on criteria over the source marketplace, over the offerings and catalogues, including available certifications, etc.) enabling control of which offerings can be exposed to local consumers. Whenever a new offering (or any update to it) published on the source marketplace matches replication and acceptance policies, it becomes automatically visible on the dome central marketplaces, with no manual action or unneeded delay (other than latency and replication times).
2. a notification mechanism enabling the DOME operator to be aware of new offerings wishing to be published in the DOME marketplace, so that it can accept/reject them individually/massively and/or set/update acceptance policies.

5.3.5.3 How policies are disseminated in the ecosystem

As introduced, replication and visibility policies are managed by marketplaces and providers as part of their catalogue and offering management. For their effective actuation within the federation, policies need to reach those marketplaces where replication is desired. To this goal, two approaches have been considered within DOME:



1. extend the TM Forum API information model for product offerings, so that replication and visibility policies are embedded in the offering itself. This approach has the advantage of simplicity, since no dedicated APIs and services need to be put in place; policies are spread in the ecosystem along with the offering itself and, when received by a marketplace, the latter already has enough information to decide about its publication. On the other hand, this approach is mostly suitable when the policies are not expected to change after their initial setup; since offerings are hashed in the DPL, any change to them (including a change to the replication policy) would require creating a new version of the offering. As soon as policies are updated, there will be a proliferation of versions for offerings that actually haven't changed in their core metadata (services, resources, descriptions, prices, agreements, etc.)
2. manage replication policies as first-level entities and advertised in the federation through DPL notification mechanisms. Although this approach requires dedicated entries in the DOME information model and a specific API for publication and retrieval, it provides more flexibility and enables decoupling between offerings and policies. Following the very same approach as other TM Forum entities (e.g. offerings, orders, usage records), policies are exposed by marketplaces in their local storage and advertised through DPL notification mechanisms. Federated marketplaces can subscribe to them and get informed about any available replication request. DPL hashing will guarantee integrity of exposed policies, and versioning would support their lifecycle. Furthermore, being documents independent from offerings, individual policies can cover an arbitrary set of offerings or even entire catalogues, possibly using expression and rules rather than just statements; also, by using rules and wildcards, policies might apply to future offerings as well, reducing the need for their update over time.

5.3.5.4 Enforcement of policies

Replication policy enforcement is about ensuring that the desiderata expressed by marketplaces/providers is correctly applied by federated marketplaces. As already mentioned, marketplaces retain full control on what to expose on their catalogues but, if they do, they must comply with the owner's expectations. This means that:

- they cannot expose any offering that the owner is not willing to replicate.
- if they accept to expose an offering, this has to be done in full respect of the established policy.

The approach, based on the above principles, relies on two enforcement points:

1. A first enforcement point is the source marketplace/provider. In fact, according to the replication policies it defined itself, it can forbid access to any offering/catalogue which is not allowed by established policies.
2. A second level of enforcement happens on the replicating marketplace and concerns the visibility rules (e.g. selective disclosure of offerings or part of them to different consumers, according to their attributes, authentication, etc.). Here the marketplace should implement, as accurately as possible, the visibility desiderata. At this stage, governance and trust among stakeholders play an important role since there cannot be a technology-wise guarantee, for the owner, of accurate enforcement by the exposing marketplace.



5.4 Service Brokerage

5.4.1 Overview

The previous sections of this chapter have introduced some capabilities of the DOME ecosystem that, although complying with the DOME specifications, are mostly delivered by individual marketplaces including the DOME central one.

In a basic relational scenario, the DOME central marketplace provides a trusted and comprehensive catalogue of cloud services, enriched with available certifications, price lists and communication links with the various providers. Consumers are enabled to search for cloud services and contact providers with the support of DOME, but subsequent steps in the transaction (ordering, provisioning, payment, etc.) take place on the providers' facilities, without any involvement of the DOME marketplace.

DOME goes beyond this model and aims at delivering transaction-facilitation features. In this transactional scenario, the marketplace model evolves to incorporate brokerage services like a federation-aware order management system. Although the delivery of complex and distributed services will not be in the scope of the project, related business processes like consumption tracking, billing, invoicing and payment will be supported by DOME with the goal to ease the procurement process for consumers and seamlessly increase visibility and revenues for providers and federated marketplaces.

The remainder of this section will cover the features mentioned above, analysing the different possible scenarios in the DOME federation, reporting on major design choices both in terms of federated processes and of structure and behaviour of the DOME services delivering those features.

5.4.2 Order Placement

Order placement is the process of submitting an order for cloud services to a cloud marketplace. The order placement in non distributed scenarios involves the following steps:

- The customer selects the cloud services that they want to order.
- The customer provides the necessary information about the order, such as the quantity of each cloud service, the duration of the order, and the billing information.
- The customer submits the order to the cloud marketplace.
- The cloud marketplace forwards the order to the relevant cloud providers.
- The cloud providers execute the order and provide the cloud services to the customer.

The order placement process can be automated or manual. In an automated process, the customer submits the order to the cloud marketplace through a web portal or API. The cloud marketplace then automatically forwards the order to the relevant cloud service providers. In a manual process, the customer submits the order to the cloud marketplace by phone, email, or fax. The cloud marketplace then manually forwards the order to the relevant cloud providers.

We are targeting automated order placement in a federated environment leveraging the IAM and a distributed storage infrastructure (described earlier in this document). Thanks to the infrastructure underneath, actions performed locally will be automatically replicated over the



federated marketplaces detaching each of them from the burden of taking care of the DOME cloud ecosystem.

5.4.2.1 Federated Scenarios for order placement

Service offerings are published on marketplaces (DOME and federated marketplaces) according to their internal procedures. Federated marketplaces should advertise the existence of new offerings by creating an entry in the DPL (TMF APIs). However, not all services are eligible to be part of the DOME catalogue/ecosystem. DOME verifies service compliance against the necessary minimum certifications and, once all checks are successfully passed, publishes it in the DOME catalogue. Only services approved by DOME can be sold in the EU. Federated marketplaces must take the services in the list of those respecting a minimal set of rules.

DOME, like other federated marketplaces, can decide whether to sell the services published or simply keep them in the catalogue for customers to browse.

The DOME governance will establish how DOME will manage services that are not compliant with EU rules. Those that cannot be sold in the EU will be marked in the list to be clearly distinguished by users. The WG working on D5.1 will provide guidelines to be discussed within the ecosystem.

As mentioned earlier, the DOME central marketplace can act as a selling channel for product offerings created and published locally (by the DOME marketplace itself) and, most prominently, as a broker between consumers and federated marketplaces that decide to expose their offerings in the federation. D5.1 will propose ecosystem rules on how to sell through the DOME marketplace.

Since simply forwarding to a federated marketplace would make it impossible to track the sales of services offered by different marketplaces, the DOME executive board will establish appropriate ecosystem policies. This could be the case for services that cannot be sold in the EU. It could also be decided that all services in the catalogue must be sold by DOME itself or with its intermediation.

In conclusion, the DOME central marketplace can act as a selling channel for product offerings created and published locally and, most prominently, as a broker between consumers and federated marketplaces that decide to expose their offerings in the federation. The DOME ecosystem will be ruled by specific business and governance policies.

5.4.2.1.1 Placing orders on DOME for offerings linked to a Federated Marketplace

When the DOME central marketplace acts as broker, two scenarios might occur when it comes to order placement.

The relational scenario

In the *relational scenario*, no ordering happens on the DOME central marketplace. Instead, for all published product offerings in the DOME central marketplace, the user will be redirected to the Federated Marketplace for order placement (including contractual customization, configuration of product options, etc.). From this moment on, the DOME central marketplace won't have any further intermediation in the order processing workflow.

DOME will support the relational scenario in its basic feature set. The DOME ecosystem rules will define the utilisation scope of this mechanism.



For the sake of simplicity, the diagrams below show direct interactions between marketplaces, as this is the focus of this section. However, from a more technical perspective, every relevant interaction within a transaction (e.g. order placement, status update notification, etc.) is tracked using DOME mechanisms and notarized in the Decentralised Persistence Layer.

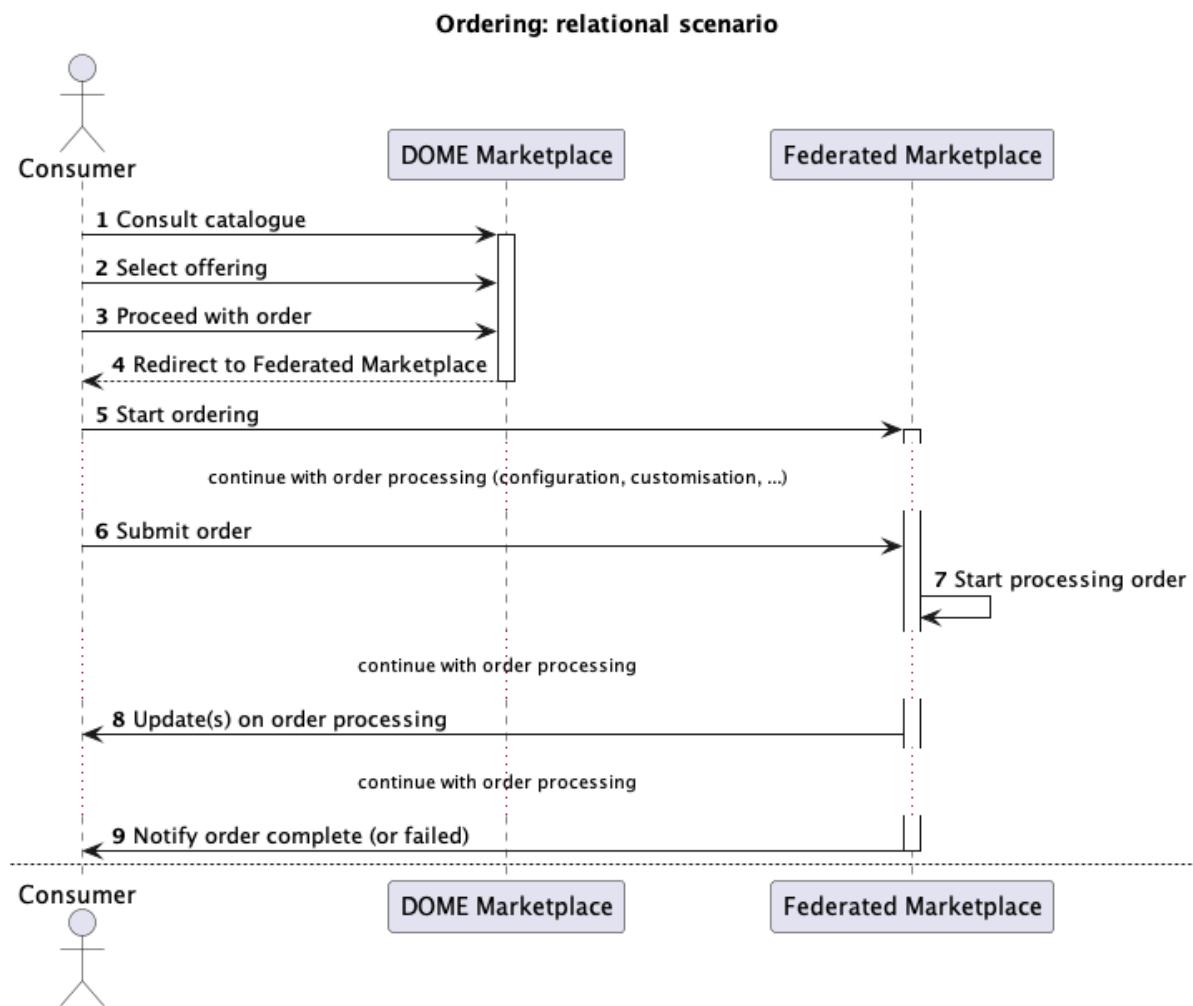


Figure 44 - Ordering: relational scenario

The transactional scenario

In the *transactional scenario* supported by DOME, service providers whose offerings have been published in the DOME marketplace can decide to exploit ordering facilities available there, leaving the burden of order management (e.g., cart management, order definition, configuration of options, contract customization and signature, billing and payment details, etc.) to the DOME central marketplace. Upon creation of the order and its dispatch, the service provider takes over the provisioning/delivery process, while notifying the relying party (i.e., the DOME marketplace) about process progress until completion.

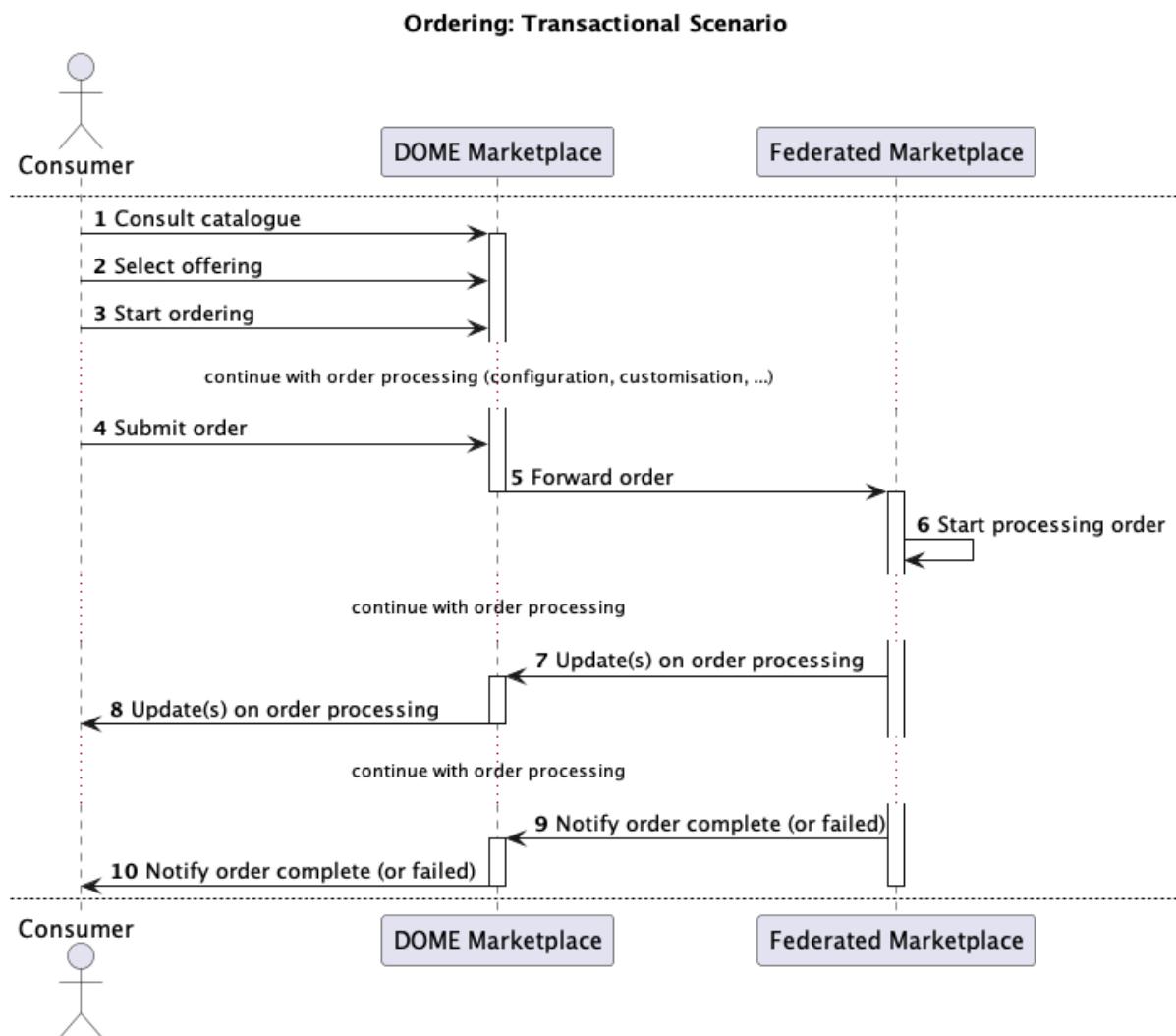


Figure 45 - Ordering: transactional scenario

It is worth noting that, in a transactional scenario, mixed behaviours will still be needed/preferred by some service providers. Depending on business choices, the service provider might choose to manage exclusively the order processing for some offerings. Also, for a given offering with complex product configuration, it might decide to let DOME manage orders for products with basic configurations and take over the process to provide the consumer with personalised user experience in the case of more advanced configurations.

5.4.2.1.2 Placing orders on DOME for offerings linked to the DOME marketplace

Clearly, when the product offerings have been originally published on the DOME central marketplace, the process does not differ from the transactional scenario, for what concerns the placement of orders; in this case the DOME marketplace will play both the roles of selling and providing marketplace, as we can see in the following diagram.



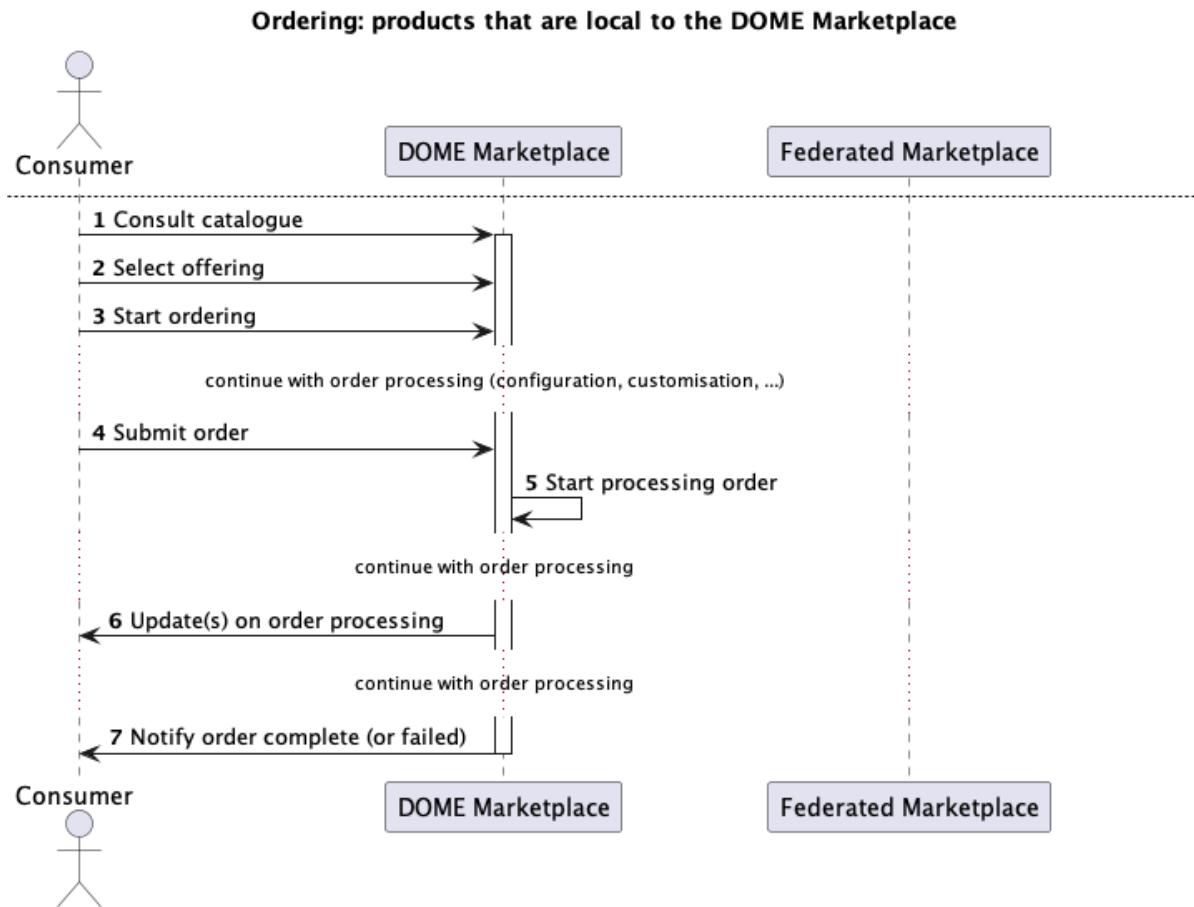


Figure 46 - Ordering: products that are local to the DOME marketplace

5.4.2.1.3 Placing orders on Federated Marketplaces

When orders are managed and placed by Federated Marketplaces for offerings published within the DOME federation (no matter whether in the DOME Marketplace or another Federated Marketplace), the order placement can fall into any of the three scenarios described above. Federated Marketplaces can freely decide to adopt the transactional or relational approach, depending on their technical and business choices.

5.4.2.2 Placing orders on the DOME marketplace

In addition to the federated scenarios, the DOME Marketplace can be used directly for the publication of product offerings, hence product orders can be placed locally in the DOME central Marketplace.

5.4.2.2.1 The product order information model

For the Management of product orders the DOME Central Marketplace will rely on the TM Forum Product Order Management API using the Product Order model. A Product Order, as defined by TM Forum, is a type of order which can be used to place an order between a Party (mainly customer) and a service provider. It is connected to a Product Offering in a Product



Catalogue and, when successfully completed, leads the creation of a Product in the Product Inventory triggering the service provisioning.

The Product Order model defines the following main attributes:

- **orderDate**: Date when the order was created
- **completionDate**: Date when the order was completed
- **expectedCompletionDate**: Expected completion date amended by the provider
- **requestedCompletionDate**: Requested delivery date from the requestor perspective
- **requestedStartDate**: Order fulfilment start date wished by the customer. This is used when, for any reason, customer cannot allow seller to operationally begin the fulfilment before a date
- **cancellationDate**: Date in which the order has been cancelled (when cancelled).
- **cancellationReason**: Reason why the order is cancelled (when cancelled).
- **state**: Lifecycle status of the order
- **relatedParty**: A list of related parties. A related party defines a party or party role linked to a specific entity. It includes the customer, provider and other parties that may be involved
- **category**: Used to categorise the order from a business perspective that can be useful for the ordering management system (e.g. "enterprise", "residential", ...)
- **priority**: A way that can be used by consumers to prioritise orders in ordering management system (from 0 to 4 : 0 is the highest priority, and 4 the lowest)
- **externalId**: ID given by the consumer and only understandable by him (to facilitate his searches afterwards)
- **note**: A list of notes. Extra information about a given entity
- **notificationContact**: Contact attached to the order to send back information regarding this order.
- **agreement**: A list of references to agreements. Agreements defined in the context of the product order
- **billingAccount**: A reference to a billing account of the customer
- **payment**: A list of references to Payment resources, as defined in the TM Forum Payment Management API
- **productOrderItem**: Each of the parts of the Product Order, including one per Product in the product inventory to be created or managed. Each product order item includes the reference to the product offering, the chosen characteristics and pricing, and the action to be performed: add, modify, delete and noChange

It can be seen that the TM Forum model allows multiple actions to be performed by the ordering management system, defined within the product order items. In practise, the model allows customers to manage the product inventory via products orders, instantiating new products with the add action, updating an existing product instance with the modify action (updating this way the terms of the purchase), and removing the product from the inventory with the delete action (terminating the contract). Nevertheless, from the context of the DOME Marketplace only add action will be supported, ensuring that a particular product in the product inventory is linked to a single product order. This approach will simplify the ordering management process and the scenarios the Marketplace will need to handle. Updates and termination of the contacts will be managed directly by the DOME Marketplace and the Inventory Management APIs applying the needed access policies.

The TM Forum model for Product Orders defines a complete lifecycle that is depicted in the following picture.



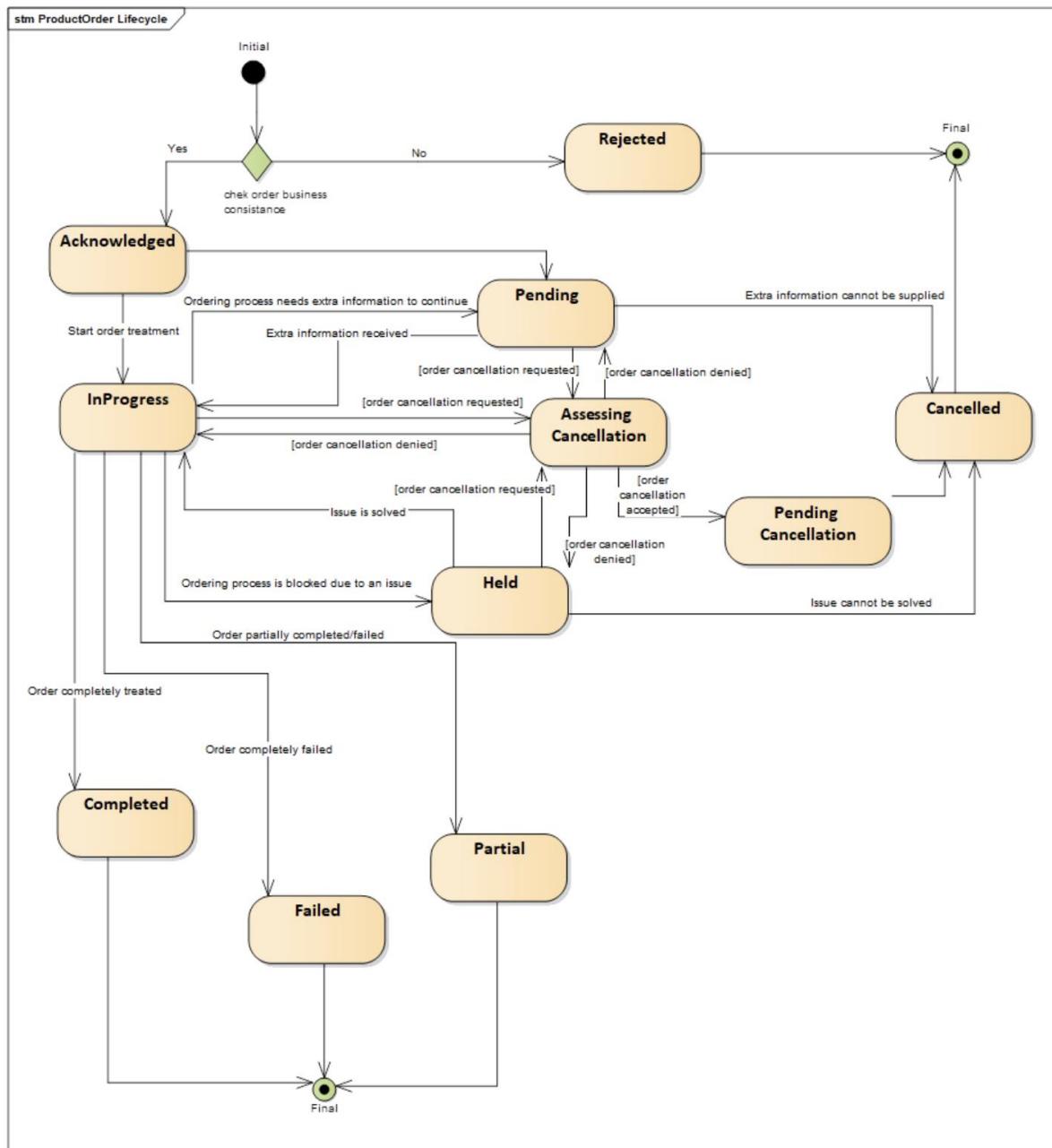


Figure 47 - Lifecycle of a product order

When a new product order is submitted, the system makes an initial validation of the information that has been provided by the customer. If such information is not consistent (i.e. the product offerings do not exist or the chosen characteristics values are not valid), the order is directly placed in the Rejected state. If the information is correct, the order is placed in the Acknowledged state so it can be processed.



When the order begins to be processed, it is set to InProgress state. From such status, if extra information needs to be provided, the order is set to Pending state until provided. If such information cannot be provided the order is directly Cancelled. In addition, if the order is requested to be cancelled, it is set to Assessing Cancellation and if accepted it is Cancelled through the Pending Cancellation state. Finally, if some issue blocked the ordering process the product order is set in Held status. In case the issue cannot be fixed the product order is Cancelled.

When the order has been processed, it can be set in 3 different states depending on how the order process has finished: (1) If all the order items have been correctly processed, the order is set in Completed state. (2) If all the order items have failed to be processed the order is set in Failed state, and (3) if some of the order items have been processed the order is set in Partial state

5.4.2.2 Configuring the product offering

The DOME Marketplace will allow customers to configure some options when placing a product order. In particular, to select the product offering price and the values of product characteristics when multiple options are available.

On the one hand, the product characteristics are defined as part of the product specification included in the product offering. Such characteristics allow them to provide custom information about the specification. In addition, the TM Forum model allows multiple values for those scenarios where customers are allowed to select one option (i.e. multiple storage sizes, bandwidth, etc). Moreover, the TM Forum model supports different pricing schemes depending on the values of the characteristics by assigning to the product offering the allowed values that can be chosen when acquiring it. With such an approach it is possible to define different offerings linked to the same product specification providing different pricing models for the different characteristics values.

On the other hand, the product offering can include multiple pricing models. In this case, the customers will be able to select the one that best fits their needs (i.e a pay-per-use model vs a recurring payment)

Those options, characteristics and pricing, are provided as part of the product order item associated with the product offering, and are used as the basis for the instantiation of the product in the product inventory once the product order is completed.

5.4.2.3 Agreement customisation

The Product Ordering Management API, as defined in previous sections, allows customers to place orders for acquiring one or more product offerings covering the basic customer-seller relationship. This scenario can be enriched in the DOME Marketplace using the TM Forum Agreement Management API for supporting any agreement between parties.

As described in section 5.3.3.3 “Agreements management”, service providers can create Agreement Specifications to define the templates of the agreements they are willing to create. Using such specifications as the starting point, it is possible to create Agreements in the Agreement Management API. The agreement model will include the reference to the agreement specification, the list of characteristics that describe the agreement, its main goal, and the dates the current agreement will be valid.



In addition, the agreement will describe the list of engaged parties that participate and the role they will play, as well as the list of authorizations from the different representatives of each of the parties, including the status of the authorization, the signature of the representative and the date of the signature. Note that for the agreement to be valid all the signatures from the different parties must be provided.

Finally, the agreement will include the different agreement items that are subject to the relationship being established. The agreement items can include a reference to a product offering and the specific terms and conditions applied to it. When the product offering is procured and the different services and resources included have been provisioned, the agreement item will include also the reference to the product in the Product Inventory Management API that references the different service and resource instances (as described in section 5.4.3.2.3).

5.4.2.2.4 Including customer and billing information

Placing product orders in the DOME Marketplace requires customers to provide the billing information to be used during the payment process. Such billing information will be managed in the DOME Marketplace by using the TM Forum Account Management API. This API provides a standardised mechanism for the management of billing and settlement accounts as well as for financial accounting either in B2B or B2B2C contexts.

Users in the DOME Marketplace will be able to create and update different Billing and Settlement accounts (if they are service providers), that can then be used when orders are placed. In particular, when submitting a new product order the customers will need to provide the billing account they want to use by using the `billingAccount` field described in section 5.4.2.2.1.

5.4.2.2.5 Submitting the order to the Decentralised Persistence Layer

When a new product order is placed, the DOME federation ecosystem (and precisely, all the involved actors) needs to be aware of such a placement in order to properly track customer actions and process the order. In particular, the DOME federation ecosystem needs to be able to manage the payment and notify the service provider that a new product order has been placed for provisioning the services, as described in section 5.4.3 “Service Provisioning”. To do so, the DOME federation ecosystem relies on the Decentralised Persistent Layer, which will store all the entities generated during the ordering process as all the interactions will be performed with the TM Forum API exposed by the access node, as described in section 4.

5.4.3 Service Provisioning

A cornerstone part of the ordering process described in the previous section is the provisioning subprocess.

Once the order is complete and there's an agreement between the consumer and the provider, the latter can proceed with the provisioning of the product, i.e. setup of services and resources defined in the product specification to serve consumer needs.



5.4.3.1 Provisioning scenarios in the DOME Federated environment

The entity triggering the start of the provisioning process is always the marketplace orchestrating the ordering process, upon its conclusion. Also, that's the same marketplace that keeps the consumer updated about the progress of the provisioning, until the offering is fully provisioned. From then on, the consumer can start using the product without any further intermediation of DOME entities.

Depending on the nature of the product offering (and/or of the service and resource candidates making it), the actual product might be provisioned and active before and independently of any order or provisioning request for it. However, still in this case, the provisioning stage perfectly makes sense as it should, at least, consist in granting enough access rights to the consumer, triggering the start of consumption tracking, etc.

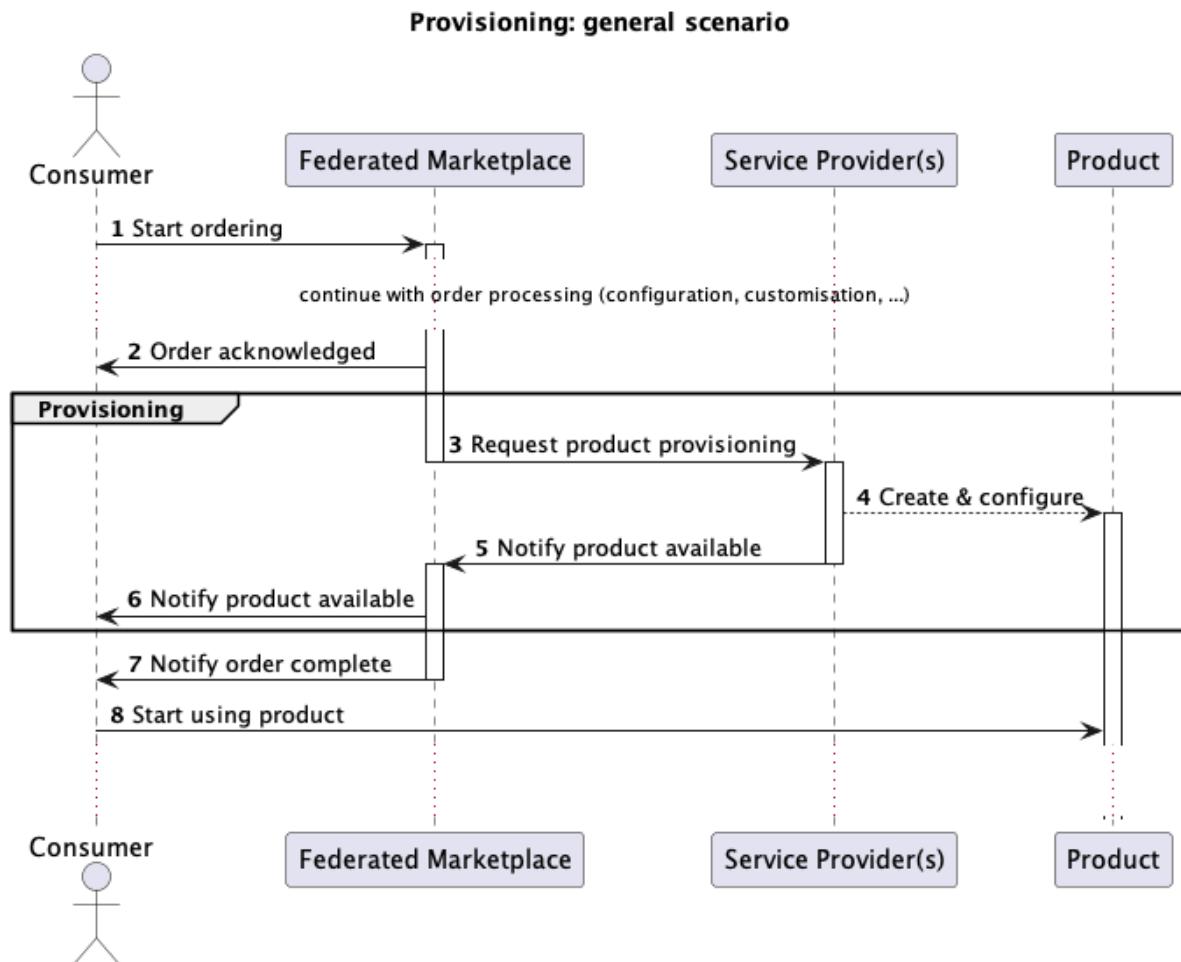


Figure 48 - Provisioning: general scenario

The scenario depicted above is quite general and can occur with slight variations, depending on the actual DOME entities playing the different roles. In the following, we'll explore them individually.

For the sake of simplicity, the above diagram does not show actions involving the DOME Decentralised Persistence Layer.

5.4.3.1.1 Federated Marketplace and Provider linked with a legacy integration

In this scenario, there's usually a business and legacy technical link between the Marketplace and the Provider, established outside the DOME federation. In terms of provisioning, it's the Marketplace that holds the burden to comply with DOME specifications implementing the Provisioning API, whereas the interaction between the Provider and the Marketplace is either based on a legacy API provided by the Marketplace or, the other way round, realised by absorbing the provider specific API into the Marketplace. The relevant point here is that the Provider, from a technical perspective, doesn't have a direct interaction in the ecosystem regarding the provisioning process; it's not the provider that receives the request from the Access Node, nor provides updates on provisioning progress. Instead, the provider updates the marketplace about progress in a legacy fashion. The marketplace, in turns, notifies interested parties via Events and TM Forum APIs

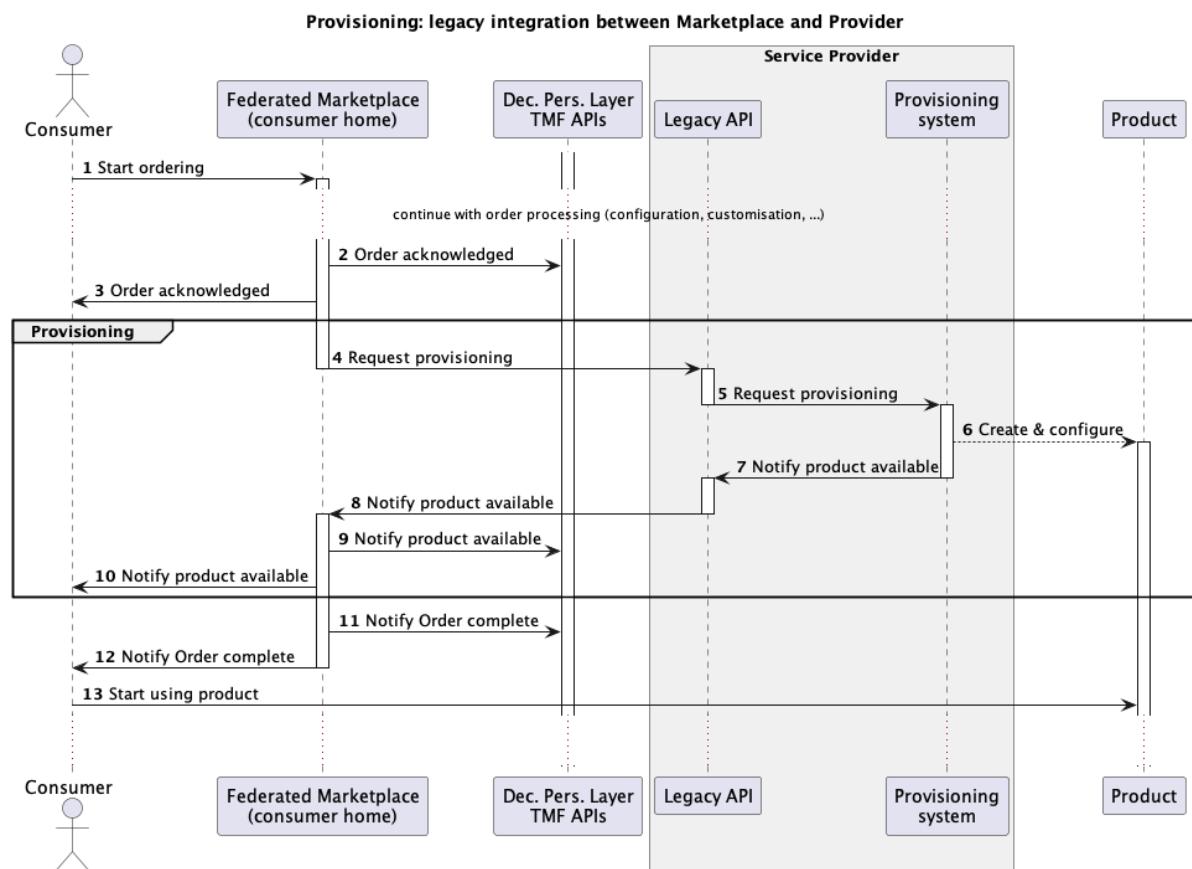


Figure 49 - Provisioning: legacy integration between Marketplace and Provider



Depending on the number of involved providers and the identity of the marketplace operator, we can find here both the 'eCommerce' scenario (where a Provider runs its own Marketplace) and the 'independent marketplace' scenario (where many providers rely on an external marketplace to sell their products).

In the following diagram we can notice that, although the interactions among entities are unchanged, the Marketplace and the Provisioning system are both under the responsibility of the Provider.

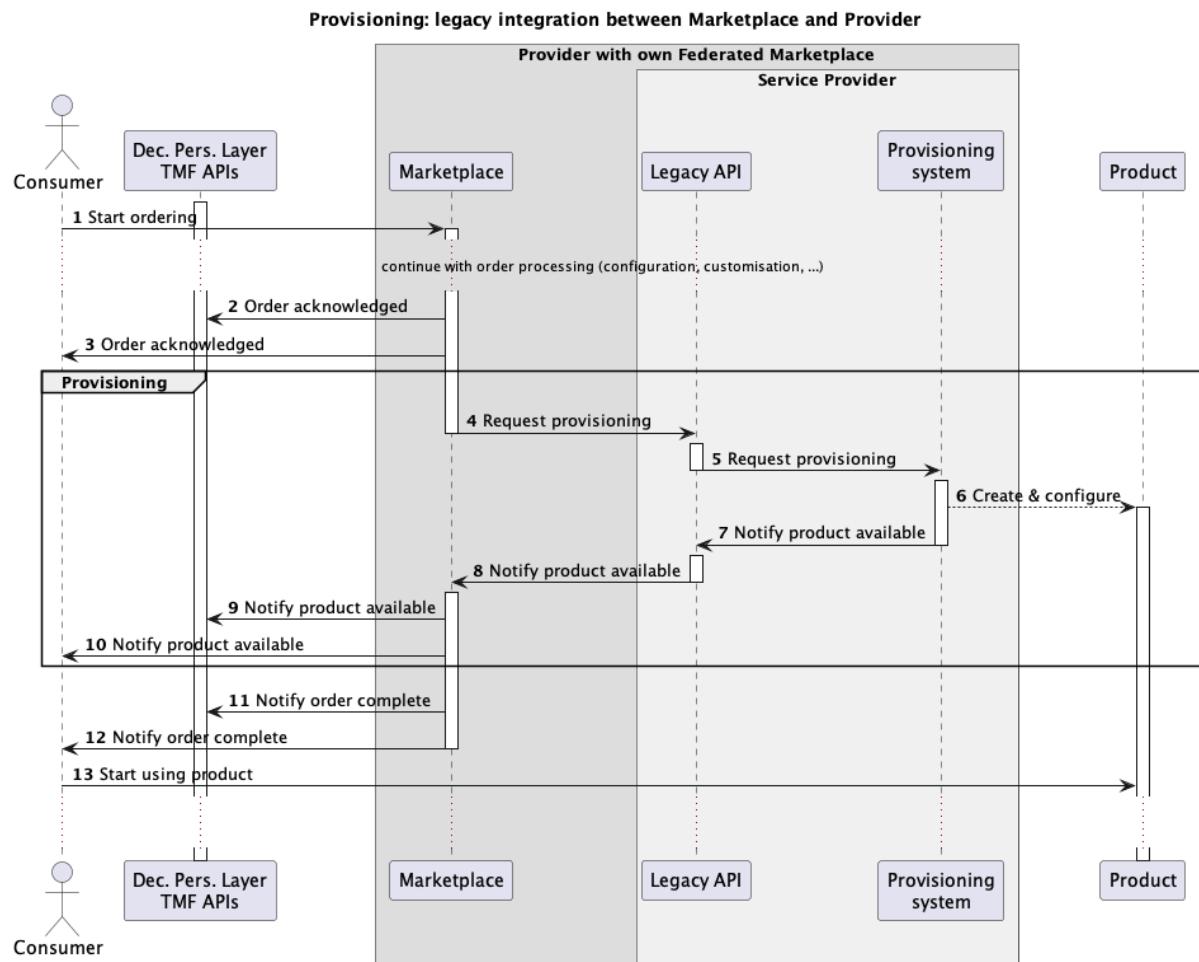


Figure 50 - Provisioning: legacy integration between Marketplace and Provider

Note: the 'eCommerce' scenario is also the case of those DOME providers that will be adopting the BAE with its plugin-based mechanism to expose their own offering in the Federation.

5.4.3.1.2 Federated Marketplace and Provider linked through DOME

In this scenario, the technical link between the Provider and the Federated Marketplace is based on the DOME technical specifications for the provisioning process. This setup brings



clear advantages in terms of interoperability; the provider is only required to implement DOME APIs, gaining the potential to join directly any marketplace in the federation. Similarly, marketplaces grounding on stable and widely-accepted provisioning APIs will lower the barriers to providers wishing to enter into business with them.

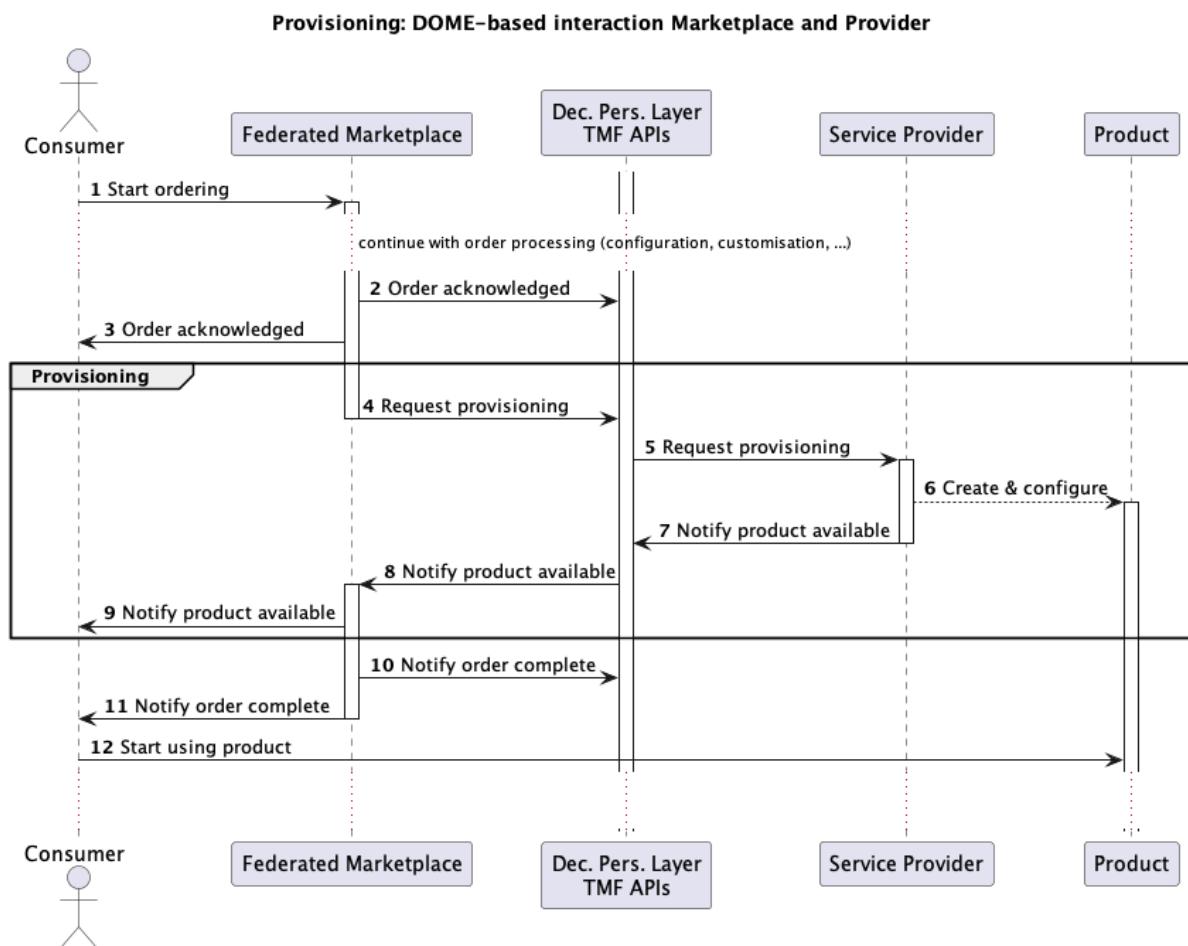


Figure 51 - DOME-based interaction between marketplace and provider

As in the previous section, the technical approach is theoretically applicable to both the 'eCommerce' and 'Independent Marketplace' scenarios.

Although this technical approach brings major benefits to the 'Independent Marketplace' scenario for what concerns interoperability and notarization in the DOME DPL, the 'eCommerce' setup can also take advantage of the adoption of standard interfaces to support diverse future scenarios.

A notable instance of this scenario is represented by the DOME Marketplace (actually one of the many Federated Marketplaces) who will allow Providers without any own/linked marketplace to publish their offering and manage orders, provisioning, tracking, etc. In order to be vendor-neutral, DOME shouldn't support any legacy integration with providers (the case



described in the previous section); instead, it should only onboard providers with DOME-compliant provisioning APIs.

5.4.3.1.3 Different Selling and Providing Marketplaces

In this scenario, the consumer finalised the order on a marketplace other than the one linked to a given product/provider. This might happen because federated marketplaces exploited DOME replication features to increase offering visibility in the federation. Depending on how the replication is configured, the consumer might be able to discover and buy a product on the marketplace he's more familiar with.

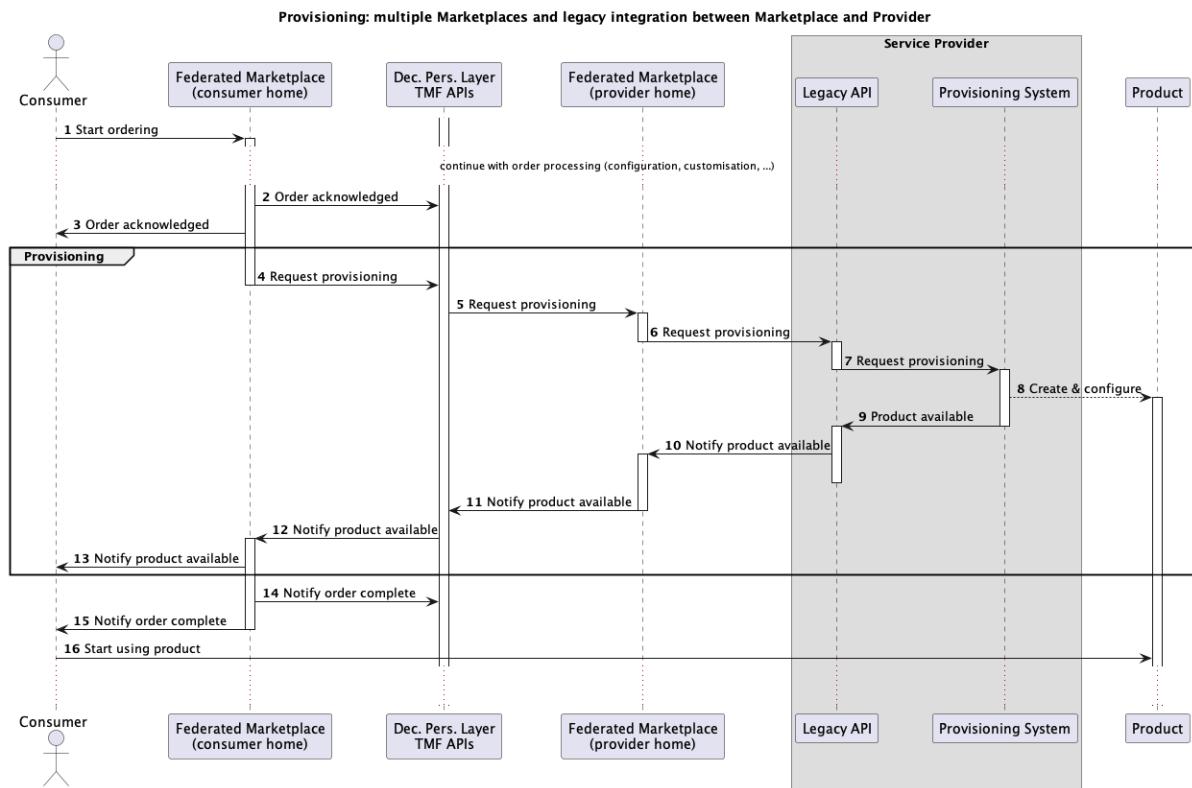


Figure 52 - Provisioning: multiple marketplaces and legacy integration between marketplace and provider.

It can be noticed that, apart from the interaction between the two marketplaces, the flow is the same described in the previous section.

There are a couple of variants to this scenario. It can happen that the provider exposes a DOME-compliant API for service/resource provisioning and such an endpoint is provided in the product specification, in place of that of the marketplace. In this case, the provisioning request will reach directly the Provider's provisioning endpoint without any intermediation of the marketplace (this does not prevent the originating marketplace from being involved in other processes, i.e. ordering, revenue sharing, payment, etc. or receiving relevant notifications, i.e. step 9 in the diagram below).



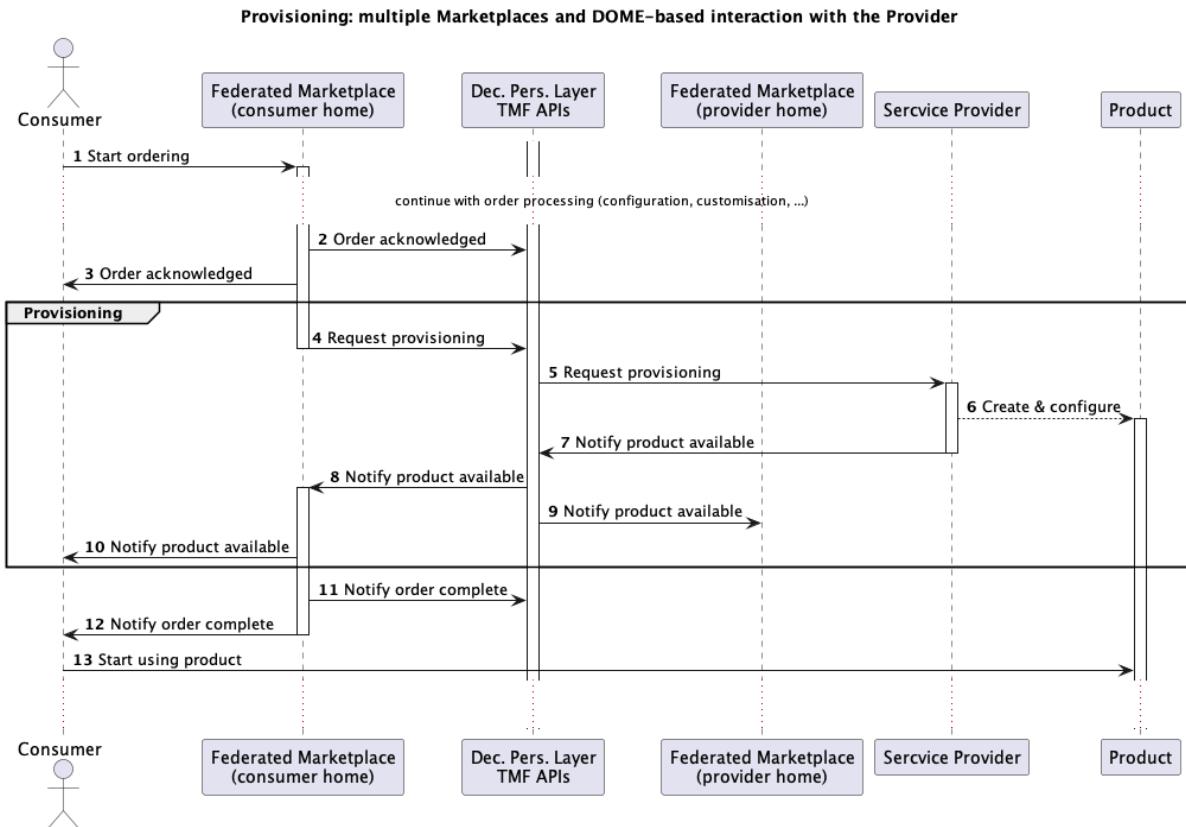


Figure 53 - Multiple marketplaces and DOME-based interaction with the provider.

Once the Provider is compliant with DOME specifications, the first scenario in this section can be revisited by introducing a DOME-aware interaction between the home marketplace and the provider.



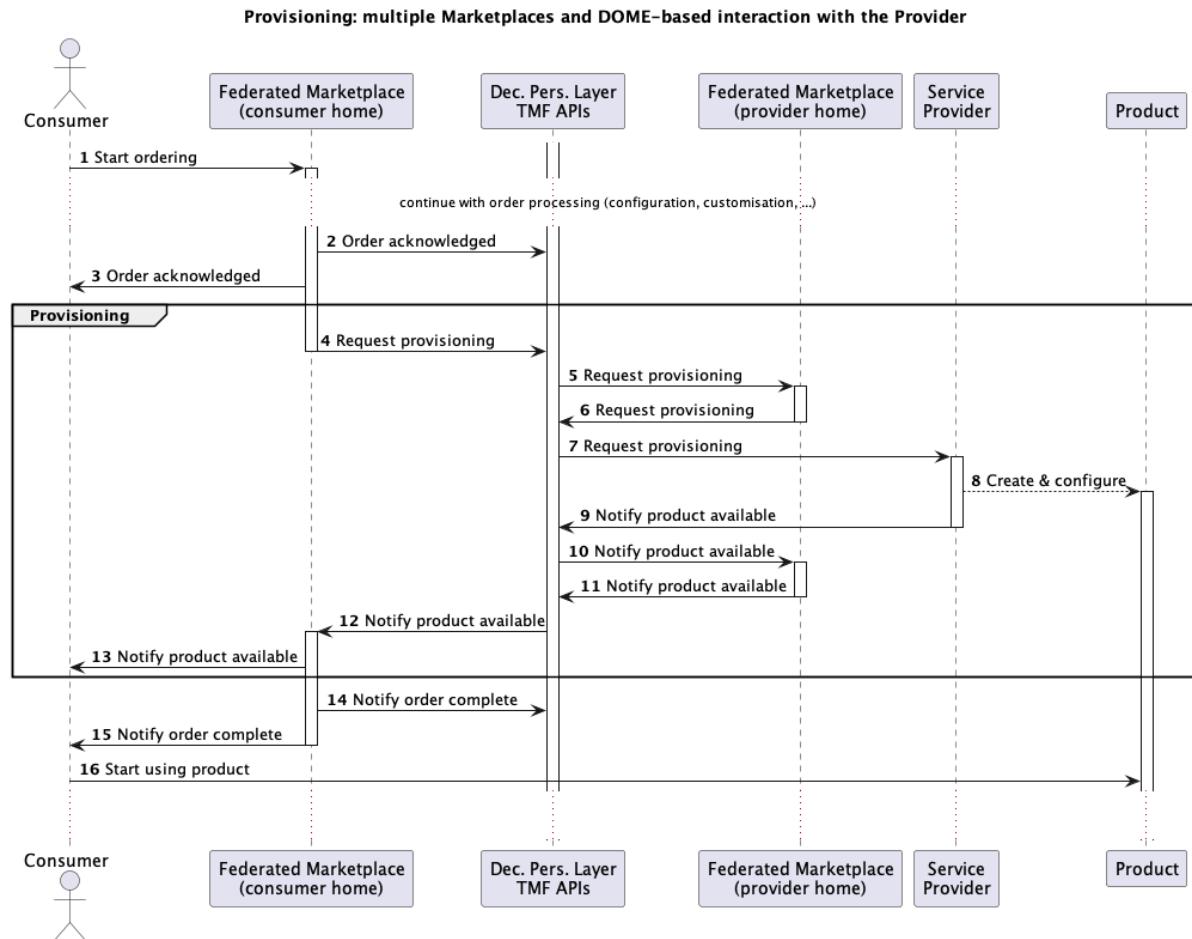


Figure 54 - Multiple marketplaces and DOME-based interaction with the provider.

5.4.3.2 The Provisioning Subsystem of the DOME marketplace

To provide access to the different services and resources offered in a product offering, the DOME Marketplace will allow different mechanisms for service and resource provisioning depending on the scenario. This section covers the different options available and describes the inventory models used to store service and resource instance information in the decentralised persistence layer.

5.4.3.2.1 The DOME-compliant Provisioning Subsystem

When using the DOME Marketplace or any of the federated Marketplaces not directly integrated with service provider legacy systems, placing orders will require the service provider to be notified so the provisioning process can be performed. That request will be sent through the decentralised persistent layer using the TM Forum APIs.

As can be seen in previous sections, after a new product order has been placed and acknowledged, the DOME Marketplace (or federated Marketplace) will request the different services involved to be provisioned. That request will be performed using the TM Forum Service Activation and Configuration API providing all the information required to instantiate



the different services (and its attached resources) taken from the product order. The request made to such an API through the decentralised persistent layer will be taken as the notification required by the service providers to provision its services and resources.

5.4.3.2.2 The Plugin-based Provisioning Subsystem

For those scenarios where service providers are willing to create their own federated Marketplace, the FIWARE BAE GE can be used as the basis of such a Marketplace. This software will be already integrated with the decentralised persistent layer through the TM Forum APIs and provides the means for integrating with legacy provisioning systems using its plugin features.

The BAE core software deals with such features that are common to any Marketplace scenario, including products and offerings management, ordering billing, etc. Nevertheless, there are multiple tasks that need to be performed during the lifecycle of a product offering that highly depends on the type of services that are being offered, including validation and provisioning. To deal with such tasks, the BAE provides a plugin-based system that allows to implement different handlers that are called during the product offering lifecycle of a specific type of service being offered. In particular, the following main handlers can be implemented:

- **On pre product spec creation:** Called by the BAE before the product specification is submitted to the TMForum Catalog API, so the specific information and permissions can be validated.
- **On post product spec creation:** Called by the BAE after the product specification has been created in the TMForum API.
- **On pre product offering creation:** Called by the BAE before a product offering is submitted to the TMForum Catalog API, so the pricing models can be validated
- **On post product offering creation:** Called by the BAE after the product offering has been created in the TMForum APIs
- **On product acquired:** Called by the BAE when a product offering has been acquired, so the service provider can provision the services
- **On product suspended:** Called by the BAE when a product must be suspended (contract termination, missing payment, and so on), so the service provider can deactivate the provisioned services.

From the point of view of DOME service providers with a legacy subsystem for service provisioning, the BAE plugin feature can be used for integrating DOME and their systems. To do that, the service providers can implement a plugin for their services, including a client for their legacy systems as part of the service provisioning handler of the plugin (on product acquired).

5.4.3.2.3 The Product/Service/Resource Inventory Subsystem

As described in section 5.4.2 “Order Placement”, customers can place a product order intended to acquire one or more product offerings, selecting characteristics values and pricing models. The result of placing such an order would be the provisioning of the different services and resources that made up the acquired product. In order to manage the information of the different instances created during the provisioning process, the DOME Marketplace will rely on the TM Forum APIs. In particular, it will use the Product Inventory Management, the Service Inventory Management, and the Resource Inventory Management APIs.



During the provisioning, the service provider will need to allocate the resources that are needed for the operation of the acquired services (a virtual machine, a physical server, a network connection, and so on). Every time a new resource is deployed a new Resource entity will be created in the Resource Inventory Management API, including all the relevant information required for the operation (location, version, VM instance ID, disk size, etc). It is important to remark that it might not be required to create a new Resource in the Resource inventory every time a product offering is acquired. There might be scenarios where multiple customers use the same real resource, and thus the same Resource instance in the inventory.

Moreover, the service provider will need to instantiate the services that have been acquired when the order was placed so the customer can get access to them. The information of these service instances will be managed in DOME as Service entities within the TM Forum Service Inventory Management API. These Services can include any information relevant for the service instance as well as the reference of the Resources that it requires to operate. In the same way as resources, it is not mandatory to create a new service every time a product offering is acquired, as the same service instance can be shared by multiple customers.

Finally, when the different services and resources have been provisioned, the acquired products need to be procured to the customer. To handle such information, a Product entity is created in the Product Inventory Management API, as the last step of the provisioning process. This Product includes the references to the resources and services that made it up, as well as all the information that have been provided as part of the product order item, including the selected characteristics values, the selected pricing model, the payments that have been made, etc. Once the product has been created in the product inventory the product order can be considered completed.

5.4.4 Consumption and Usage Tracking

5.4.4.1 Overview

Upon completion of the provisioning process, the consumer is enabled to start using the purchased product. The consumption process involves exclusively the consumer, the service provider(s); conversely the DOME platform will never take part in such a process, neither as a mediator nor as an observer.

For certain services published in the DOME ecosystem, providers may choose to embrace a transactional approach, relying on DOME facilities to support some marketplace processes. In the previous sections we've explored catalogue replication, ordering and provisioning, for which DOME can take some active role prior to service activation.

As we'll see in the following sections, DOME can also support relevant business processes at service consumption time, like billing, invoicing and payment. For such involvement to be effective and beneficial for providers, DOME must be made aware of some relevant usage information.

Usage Tracking is the process of collecting usage information, their processing, aggregation and structuring, and their publication to interested parties for the goal of applying the corresponding charges.

Tracking the usage of services by parties is mostly an activity under the control and responsibility of the service provider who puts in place all the needed probes to accurately track usage, and manages persistence facilities to record, aggregate and persist them for charging and audit purposes.



5.4.4.2 Usage Tracking in DOME

Within the DOME ecosystem, whenever a third-party process (i.e. external to the provider) is required to exploit such usage data, it's still the service provider who retains control on what information is disclosed, its granularity, how often such disclosure is done and with which delay. Clearly, there should be enough usage data and detail to enable a proper implementation of the expected service.

From the point of view of the DOME Marketplace, usage data is managed through the TM Forum **Usage Management API** and stored as part of the DOME Persistent Layer. The Usage Management API provides standardised mechanisms for usage management such as creation, update, retrieval, import and export of a collection of usages. The API consists of:

- An information model introducing a **Usage** resource, defined as an occurrence of employing a Product, Service, or Resource for its intended purpose, which is of interest to the business and can have charges applied to it. A Usage consists of characteristics, which represent attributes of usage. The API also introduces a **Usage Specification**, which is composed of characteristics, defining all attributes known for a particular type of usage, for a given type of Product/Service/Resource
- A structure for **Notifications** about relevant events related to Usage and Usage Specifications
- A set of REST API **Operations** for the creation, retrieval, update, deletion of single Usage entities, as well as the retrieval and filtering of a collection of Usage entities.

Based on the above specifications, federated providers have the means to locally store usage data and advertise their availability to interested parties who can easily search, filter and retrieve data, according to their authorization level.



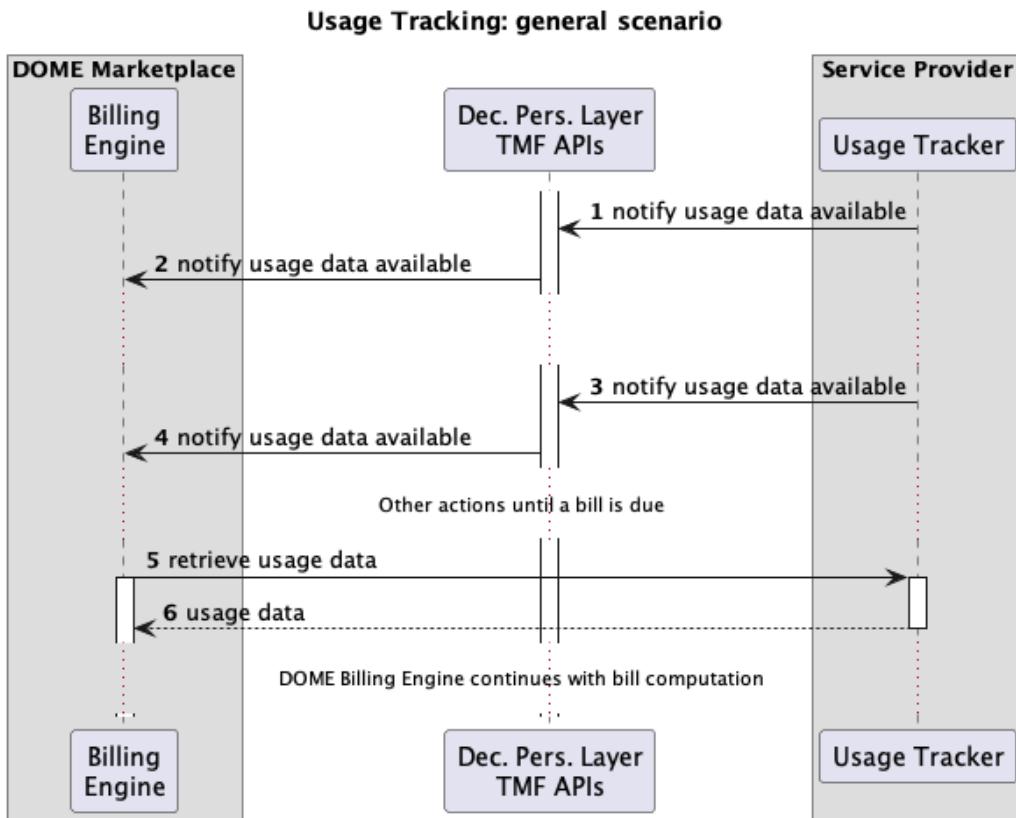


Figure 55 - Usage tracking: general scenario.

As we have already seen with Provisioning scenarios, the federated provider might not be compliant with DOME specification; rather, it relies on a legacy integration (in terms of APIs, formats, push/pull model, etc.) with its linked marketplace. In such a scenario, the party exposing the usage records to the DOME ecosystem is the Federated Marketplace acting as a broker for the Provider.

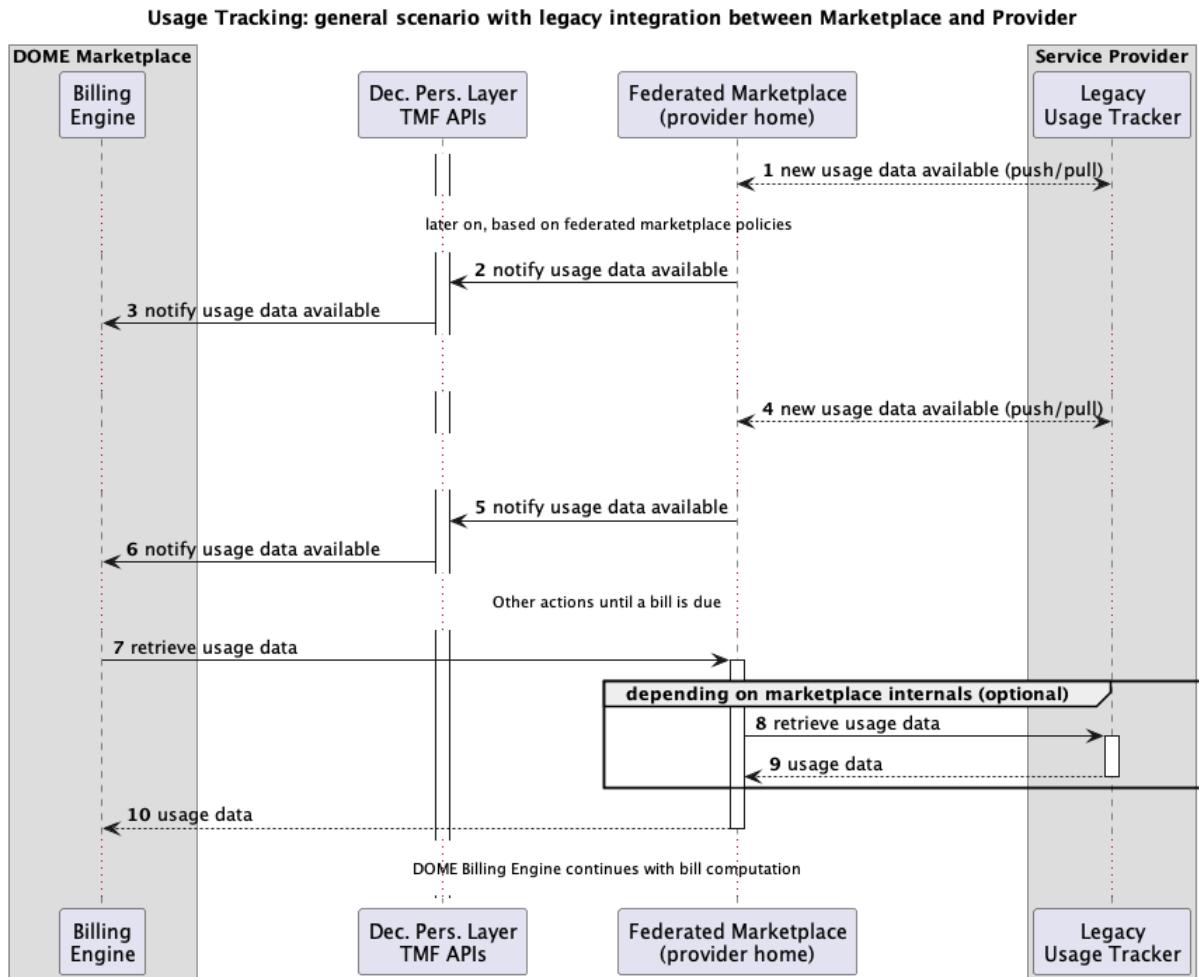


Figure 56 - Usage tracking: general scenario with legacy integration between Marketplace and Provider.

It is worth considering that, although DOME services (e.g. billing, invoicing, payment, monitoring and reporting, etc.) will be notable consumers of usage data, any marketplace willing to provide value added services can exploit the DOME technology and apply to acquire authorization and access to usage data.

5.4.5 Billing

5.4.5.1 Overview

Billing is the feature that deals with calculation of the amounts due by consumers for the purchased products. This calculation is based on pricing models that may include discount models and different charging modes (once-payment, subscriptions, pay-per-use, etc.); depending on the pricing model (e.g. pay-per-use), information about product usage is also required for a correct bill calculation.



5.4.5.2 Billing-related aspects of the DOME ecosystem

A distributed and multi-stakeholder environment like the DOME ecosystem leads to complex scenarios that need careful analysis. In particular, we can recognize the following aspects in the DOME ecosystem:

1. Product Offering bundles: offerings might consist of different independent offerings bundled together.
2. Billing and selling actors might differ: an offering might be purchased on a marketplace while the provisioning is done by a different marketplace/provider, which could also be in charge of the billing.
3. Billing computation might be as simple as in the case of a flat rate, but can also be very complex and depend on many factors. In the latter case, describing the billing algorithm in a formal way might be neither easy nor desirable.
4. For some product offerings (for example those sold on a flat rate), the marketplace/provider might decide to not expose detailed usage data.
5. Since multiple stakeholders might be involved in a single transaction (e.g. selling marketplace and possibly multiple providers/marketplaces), each stakeholder should receive a share of the revenues according to the revenue sharing policy in the product offering.

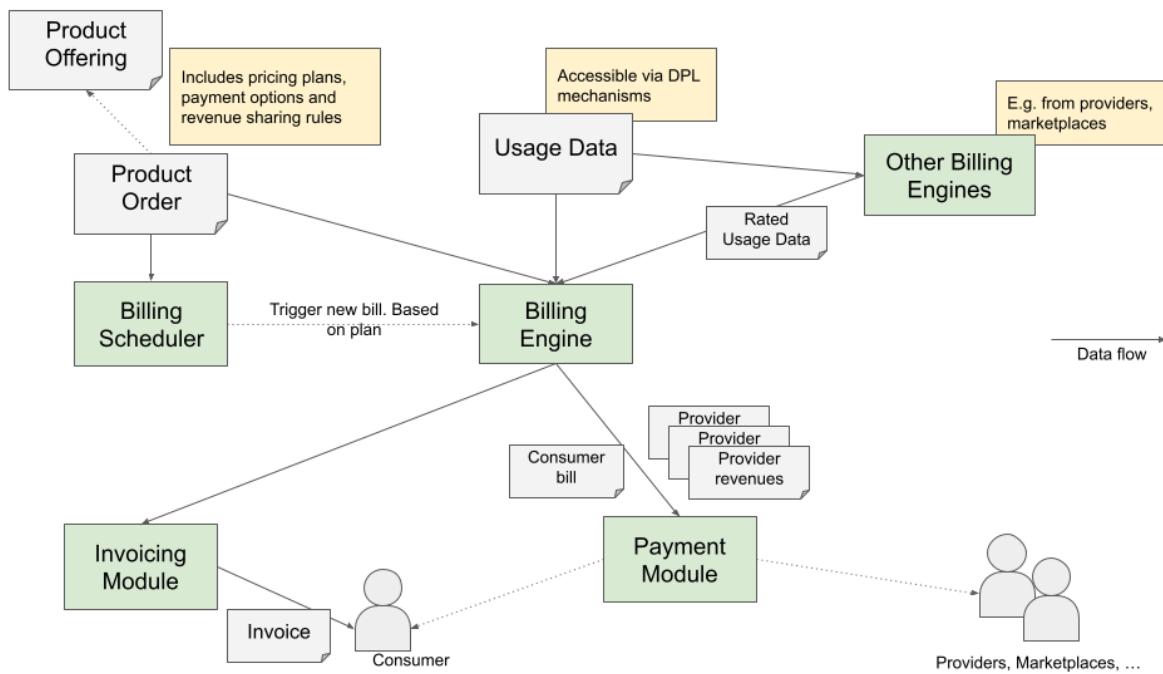
5.4.5.3 Overall approach

Given the above considerations, the following choices will apply to the design of the billing feature in DOME (some of them apply to the DOME central marketplace, others to federated marketplaces/providers):

1. A billing engine (in particular the one in the DOME central marketplace) might need different types of data to compute a bill:
 - a. A product order and related offering, as they contain the pricing plan and charging modes.
 - b. Usage data (i.e. non rated), in the case of a pay-per-use pricing model, for example.
 - c. Rated usage data (i.e. partial bills), in the case of bundled offerings, when the bills for individual sub-offerings are computed (sometimes in a complex way) by marketplaces/providers. This is also the case when the marketplace/provider only expose bills and not usage data for their offerings (e.g. when sold on a flat rate)
2. Correspondingly, a federated provider/marketplace wishing to enter a federated billing scenario should at least expose:
 - a. Non-rated usage data for a given product (enabling external bill computation)
 - b. Rated usage data for a given product (enabling external bill aggregation). As stated, this can be trivial in the case of a flat-rate plan, but also very complex; in any case this is fully under the responsibility of the federated entity and no algorithm detail needs to be exposed.
3. The billing process is triggered according to the charging modes agreed in the order and specified in the offering.
4. The billing engine (in particular its revenue-sharing computation subsystem) is also responsible for splitting the bill revenues among involved stakeholders.
5. The completion of the billing process triggers the start of the invoicing process to the consumer and the payment process (from the consumer, but also among stakeholders for revenue sharing).



On the basis of the previous considerations, the general relationships among usage, billing, invoicing and payment appear as follows. The diagram does not show the entities running the various modules (it can be the provider, the federated marketplace, the DOME marketplace, it can be a third-party service, etc.)



5.4.5.4 A deeper analysis of the Billing feature

The baseline rule in billing is the following formula:

$$\text{Unit price} \times \text{quantity} = \text{final price}$$

In a cloud marketplace environment things are a bit more complicated because the procurement process involves different items that must be considered in the right order.

To have a clear understanding of the process we can try to see the analogies with a normal shop purchase process and to do this let's try to define some reference:

- ORDER. An order is a single purchase process including different products. Any product can be different and unrelated to the others. They are part of the same order just because they have been purchased at the same moment.
- SUBSCRIPTION. A subscription is the purchase of a single product/service, including options and configuration items. A single ORDER can include multiple subscriptions. While an ORDER is a one-off operation a subscription can, upon the selling model, have recurring payments and periodical renewal process.
- PLANS. A single product/service can be sold in different ways: pay per use or flat rate, can have volume discount pricing, can have different configurations, etc. A PLAN defines a single way to purchase the specific product/service. A SUBSCRIPTION is bound to a specific PLAN selected by the customer during the ORDER processing.

- OPTIONS. Products/services can have several configuration items describing volumes, typologies, or anything else the provider considers in his offering flexibility. Such options often define the contents of the subscription.

We can describe the above in the following way:

An ORDER is the sum of several SUBSCRIPTIONS with a predefined set of OPTIONS that, mapping a specific payment PLAN, defines the payment model.

All the above describes the ordering process, not the billing. The billing in a marketplace environment is an independent process, not directly related to the order process, that, upon the order, creates a billing form (the baseline of the invoice process) calculating all the values involved in the specific transaction.

To enhance the understanding of the above sentence we need to clarify the difference between prepaid and postpaid transactions.

Pre-paid transactions are started simultaneously to the order acquisition because the customer must pay the expected fee before he receives the service. The typical case of such transactions is the purchase of the shelf products (paid once) but can be used also on "flat" recurring fees considering the payment at the beginning of the service period. Post-paid transactions are typically used for anything that is "pay per use" because you cannot bill it until the customer has used the service.

5.4.5.4.1 Scheduling

The billing process is a scheduled process. The scheduling of the single calculation is related to the payment plan associated with the subscription. While prepaid subscriptions may create a recurring (for flat) fee that can be associated to the ORDER date, it's preferable to always consider a specific day for all the calculations considering the first day of the month as standard reference and eventually consider a "special 1st month fee" calculated upon the remaining days of the first month of subscription.

5.4.5.4.2 Periodicity

While one-off payments have no specific issues, recurrent fees need to be furtherly defined, clarifying, both to the buyer and to the seller, the criteria of "periodical calculation" implemented by the billing engine.

Being most of the cloud services based on costs that are running every single day, the generic "monthly fee" that can be applied the first of every month, is normally not used, because the costs to sustain a service based on infrastructure, people and software, may be slightly different from february (28/29 days) and october (31 days) and many service provider prefers to consider a standard "30 days" as recurring period for billing.

For pay-per-use services the scenario is furtherly complicated because infrastructure providers bill most of the service on a "minute" basis. This is not related to the creation of the monthly invoice but to the need to provide real time visibility of the consumption (and related cost) of the provided services to the Customer.

While the official billing (related to the invoice process) will have a generic "monthly" recurrency, the reporting is something that needs to be calculated "on demand".

The design of the billing engine "as a service" allows the usage of the billing capability for both the reporting and the invoicing process.



5.4.5.4.3 The time series database

As described above the billing engine needs to have visibility of the consumptions of the different subscription options. That information must be provided by the provider on a continuous basis and stored in the persistent data layer for reporting and billing requirements. In the same way the billing calculation can be stored to avoid recalculating many times the same values and to speed up the reporting process.

Note: the flat rates have no “consumption information” to store, but can be anyway recorded in terms of relative billing (adding a proportional value of the consumption for the current month) to standardise the approach in the reporting function.

According with the above description we can summarise the following schema:

- The billing engine needs to have access to the SUBSCRIPTION description (the ORDER ID is not relevant for the billing calculation).
- Through the SUBSCRIPTION information the billing engine will know the list of the subscription items to be considered in the calculation, the payment type (prepaid or postpaid) and the payment model (flat or pay per use).
- For pay-per-use items it will rely on a time series database where, through the integration process (TM Forum or custom API), the provider will record the resources usage for the specific subscription.
- On a scheduled time the billing engine will calculate the cost value of the subscription and record the value on the time series database for further usage.
- Both Reporting and Invoicing engines will access the time series database with the economical value of the different subscriptions to create their own outputs.

Being the time series database accessible through DPL mechanisms, the billing information is automatically available to authorised federated marketplaces that can use it for their reporting and invoicing processes.

5.4.5.4.4 Impact on revenue sharing

All the above should be assumed to work on Customer final prices. This is because the revenue sharing is an internal process parallel to the Customer billing process approached as percentages of the final price. The revenue sharing values calculation is starting from the billing value to create the split of that value for the different involved actors for both invoicing and reporting.

5.4.5.5 Billing scenarios in the DOME federated environment

In this section we'll present some billing scenarios that are planned to be supported in the DOME ecosystem.

Scenario 1 - Product sold and billed by a single marketplace (flat rate)

The first, simplest, scenario is the one consisting of a federated marketplace exposing services from different service providers. In this simplest scenario, we assume the product offering(s) are provided by the same service provider behind the federated marketplace. Also, as stated in the product offering, the consumer is charged a flat periodic rate, regardless of the actual service usage.



In this scenario, at a given moment (it can be before and after the service provisioning, but also at regular intervals) the federated marketplace starts the billing process which will consist of trivially computing the due flat rate for the considered period.

Also, according to the revenue sharing agreement among involved stakeholders (the federated marketplace, the service provider, the dome operator, ...), a number of revenue sharing reports are also generated.

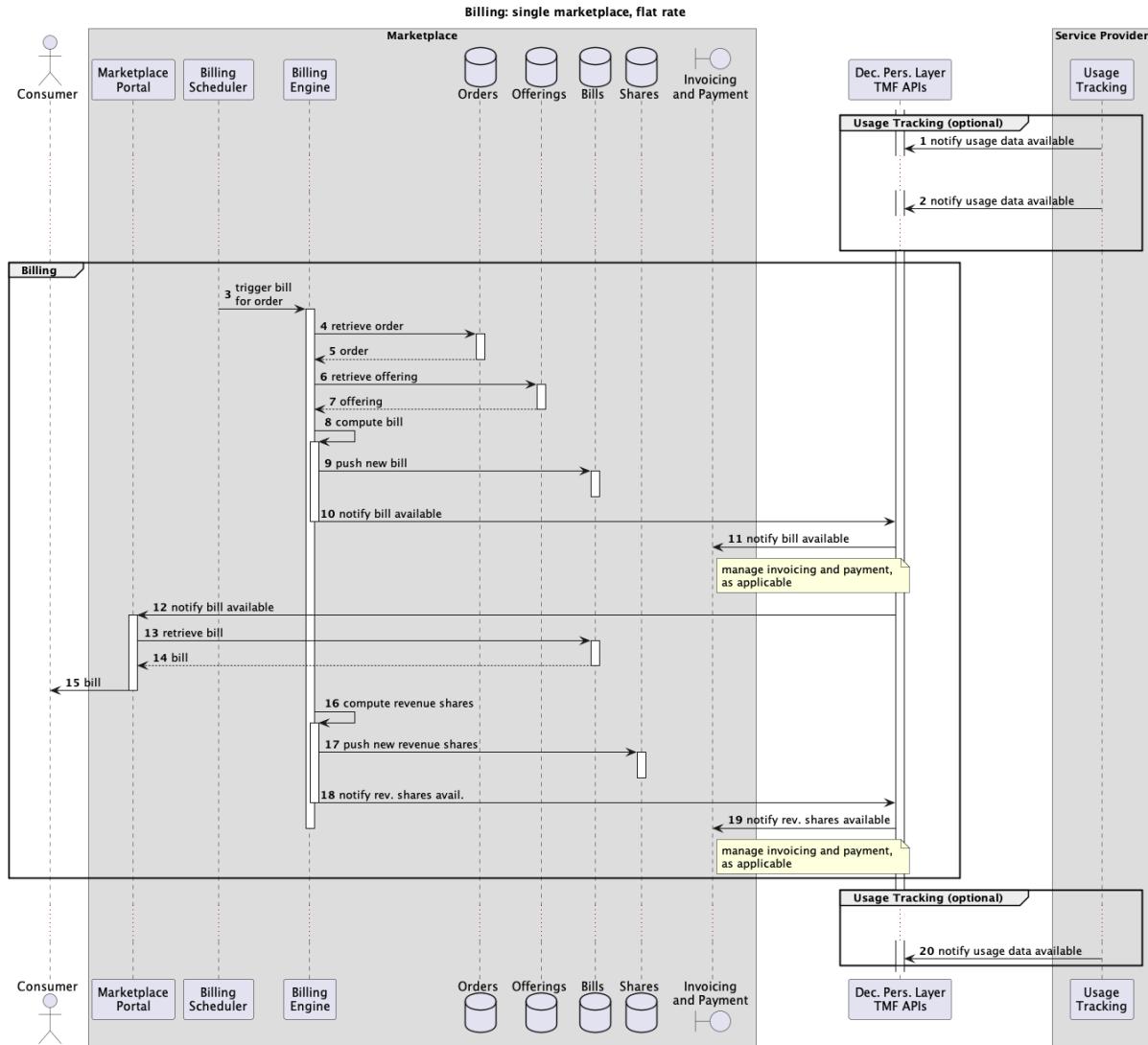


Figure 57 - Billing: single marketplace, flat rate

Depending on the approach taken for invoicing and payment, a number of further actions might be triggered afterwards or at independent moments. However, invoicing and payments are out of the scope of this section and will be considered in the following sections.

Note: although data regarding service usage are not used for bill computation, the federated/provider marketplace can still expose usage records and advertise them in the



federation through the Decentralised Persistence Layer for auditing purposes (boxes are marked as ‘optional’ in the above diagram).

Scenario 2 - Product sold and billed by a single marketplace (pay-per-use)

This scenario shares most assumptions as the previous with one major difference: the amount due by the consumer depends on the actual service usage (pay-per-use).

This difference has two implications on the billing process: first, billing can only happen after the actual service usage as opposed to the previous one when it can (and generally does) happen before; second, for the bill to be actually computed, the service provider must expose usage data (see section 5.4.4 “Consumption and Usage Tracking”). In general, the publication of usage data is an independent process on the provider side on a regular basis (either autonomously or through its linked marketplace); here we can safely assume that usage records are already available at billing time.

With the above differences, when the bill is to be generated, the federated marketplace first collects usage records and then according to the pricing policy expressed/referenced in the product offering/order, it computes the due rate for the considered period.

Note that in the diagram below, the publication of usage records is no longer optional, as it was in the previous scenario.



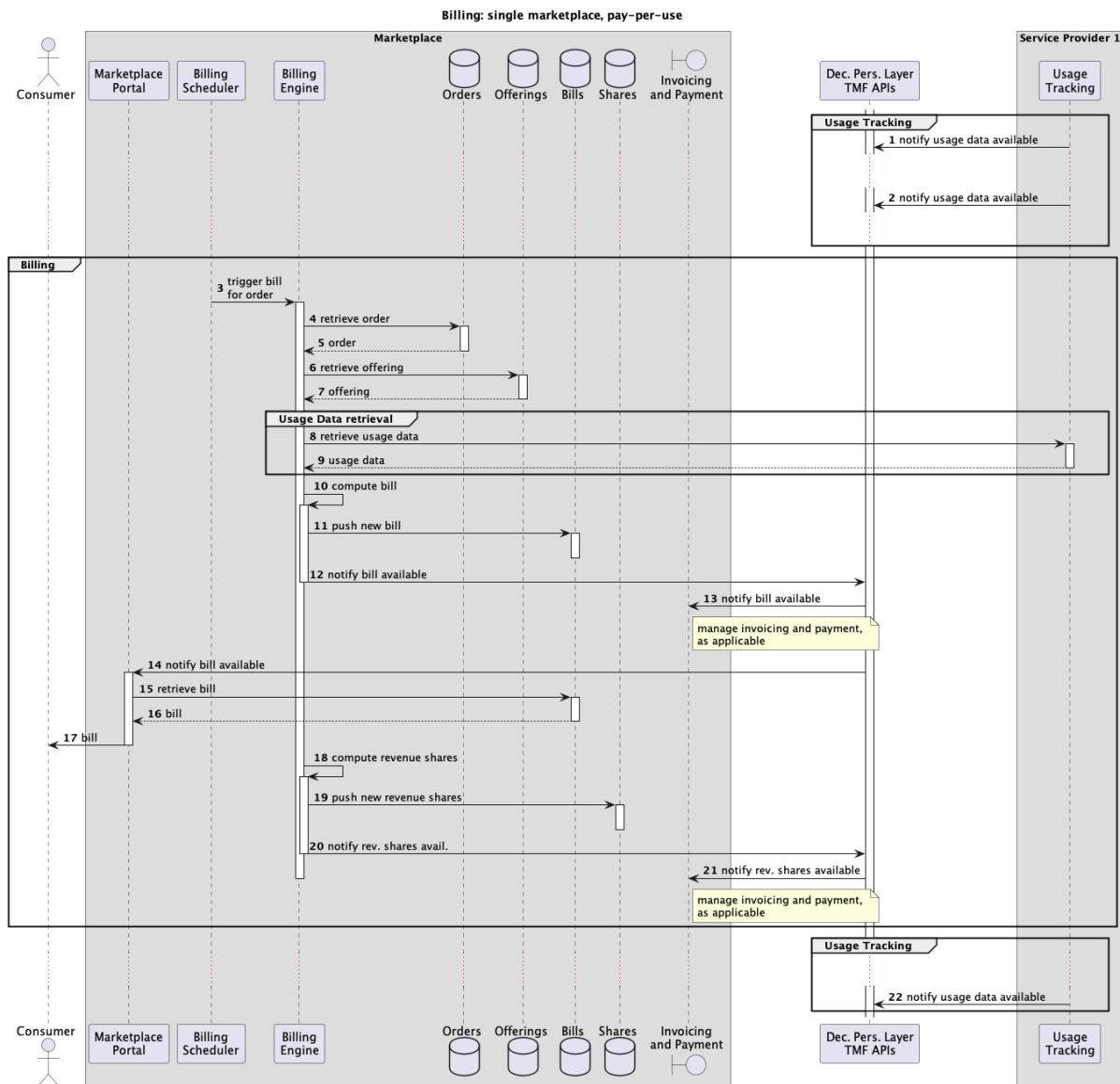


Figure 58 - Billing: single marketplace, pay-per-use.

The publication of the revenue sharing reports, triggering of invoicing and payment processes behaves as before, with no differences.

In this scenario, whenever the offering consists of a bundle of a number of other offerings, the publishing of usage data (steps 1 and 2 above) and their retrieval (steps 8 and 9) are repeated for each offering in the bundle.

Scenario 3 - Product sold by a marketplace, but billed by another one

In the scenarios above, the source marketplace of the product offering was the same where the consumer did the purchase. In DOME however, an offering/catalogue can be replicated



on so many other marketplaces within the federation enabling consumers to complete a purchase on their preferred marketplace (provided the offering is replicated there, of course).

In such a situation, the marketplace responsible for bill computation is indeed the one close to the provider, since moving this task to the selling one would mean transferring both the pricing rules (that can be arbitrarily complex and difficult to describe formally) as well as non-rated and rated usage records. In general, the approach is to compute the bill (no matter if related to a bundle or to an atomic offering) in the marketplace closest to the provider(s), where all needed information is available.

In the example below, we see a marketplace (M2) exposing a bundle BDL made of offerings O1 and O2 provisioned by a given provider. The bundle BDL is also replicated on marketplace M1, where the consumer purchased it. For the reasons above, the billing is done by marketplace M2, based on data exposed by the provider. The selling marketplace M1 will be notified of the bill through DPL events and will eventually access it, if allowed; this is normal behaviour within the DOME ecosystem and not specific to this scenario.

In the diagram below, interactions with the *invoicing and payment subsystem*, just after the publication of bills and revenue sharing reports, have been intentionally left out. Depending on the approach taken (see following sections), invoicing and payment might be managed by Marketplaces M1 or M2.



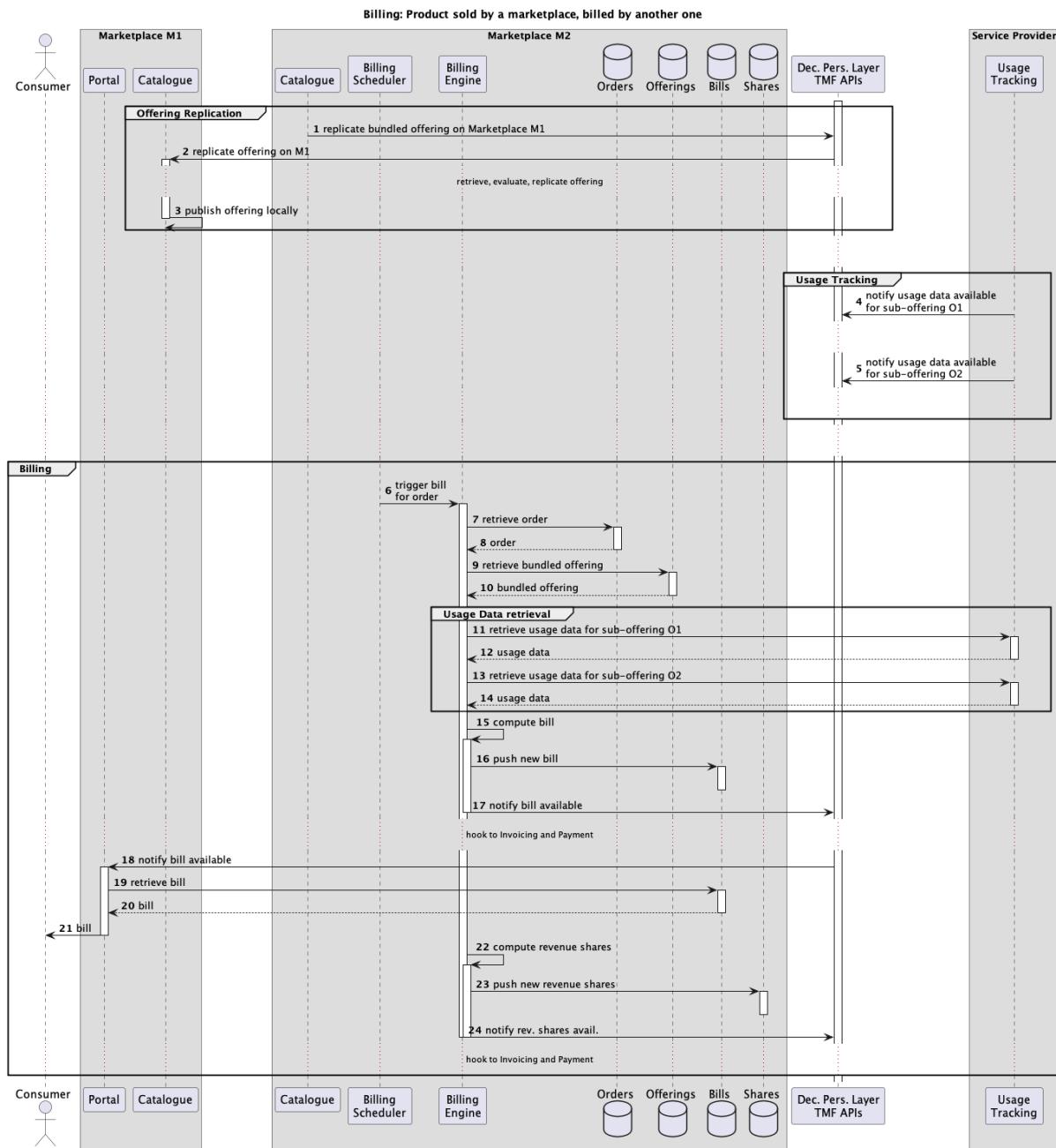


Figure 59 - Billing: product sold by a marketplace, billed by another one.

5.4.6 Revenue Sharing

Under the general definition of “revenue sharing” we’re covering all those actions referring to the capability to split the revenues related to a specific economic transaction through multiple actors. This practice is normally performed in all the “reselling” contracts where the reseller retains an agreed percentage of the bill as compensation for his selling activities.



5.4.6.1 Overview

The DOME ecosystem is a multi-entity organisation aimed to develop business. This means that different actors, with different roles will gain some revenue upon their contribution to the business model.

Like in any other business organisation we will have different “actors” participating to the business:

- Service/product providers publishing their own offer on the DOME network and aiming to gain revenues for the selling of their goods.
- Marketplace owners that will sell the services/product created by providers to the end users and aim to have some revenue to this service.
- DOME operator that enables the technical relationship between the different actors through his platform and value added services, and aims to have some revenue for that.

The revenue sharing policy is based on following main concepts:

- Providers are the owners of the products/services published on the catalogue,
- Revenues are provided upon active (and proportional) contribution to the business process.
- Some actors can cover multiple roles in the selling chain (ex. being both provider and seller) and the revenue sharing will work according to this model.
- The revenue sharing model must balance profitability between the different actors and sustainability of the offer (final price) on the market.
- Revenue sharing will be defined as a recharge % upon a baseline price defined by the product/service provider. This percentage is defined in the contractual agreement signed by the provider when he asks to be part of the DOME ecosystem.
- The contractual agreement (%value) can be revised on a yearly basis according to the subscription renewal.
- The contractual agreement will include a specific commitment to avoiding price dumping by the different involved actors.

The following sections will try to firstly address the technical architecture in order to provide all the needed functionalities to manage the “revenue sharing” process, leaving some open points on “the way” we will configure them to the Business Model decisions.

5.4.6.2 The reselling process

To technically define the “reselling” process we need to clearly understand how the reselling contract between the product owner and the reseller works. In general the product owner defines the product offering (including the description, the product options, the product component, the different selling plans etc.) The product owner also defines what we usually call the “wholesale price” that is the minimum price he can sustain to ensure his own profitability. Then the product owner defines the possible “selling price” that is something that is above the “wholesale price” as the price that the customer will pay to have the product/service. The difference between the “selling price” and the “wholesale price” is the amount of money that can be retained by the reseller to repay his “reselling” activity.

To simplify the process the difference between “wholesale price” and “selling price” is generally expressed as a percentage of the value according with the formula:

$$\text{wholesale price} + \text{margin} = \text{final price}$$



In general the margin is described as a percentage of the final price, not of the wholesale price. Assuming that the product owner will define the wholesale price and the margin percentage recognized to the reseller, the reference formula will be the following:

$$\text{final price} = \text{wholesale price} / (1 - \% \text{margin})$$

The first operation to enable the “reselling functionality” (and in parallel the “revenue sharing” model included in that approach) is to start describing in the “price list” of the BAE catalogue the difference between “wholesale price” and “final price” or better, to define a “wholesale price as baseline” and a margin percentage to be applied to define the final price.

5.4.6.3 The integration of the reseller process in the DOME ecosystem

To introduce the reselling process in the BAE platform we need as first step to define the RESELLER role in the platform.

A reseller is someone that is formally referred to by the product owner (the ones we usually call “providers”) as an entity allowed to resell their own products.

Please consider the following combinations:

- A Provider can have one or more resellers.
- A Reseller can be referred by one or more providers

In the DOME ecosystem the seller/reseller role will move by opportunity:

1. A provider selling the product on his own marketplace. He's acting as a reseller of himself, so no revenue sharing with anybody.
2. DOME portal selling a product on behalf of the provider. DOME portal will be the reseller for that transaction, so we have a revenue sharing between the provider and DOME
3. A federated marketplace selling products for a different provider. In this case is the FEDERATED Marketplace that can claim the role of reseller. In this case we have a revenue sharing between the provider, and the federated marketplace.

According with the above description we can define, at platform level, if we want to define DOME as default reseller for everybody with a predefined margin percentage to be applied on the wholesale price or if we want to have a more flexible approach to define where to have a margin and where not or which kind of margin to be applied for some product category and what else for others.

A possible option to improve the DOME operator revenues is to add the DISTRIBUTOR role that is someone that is in the middle between the product owner and the reseller.

In this case the selling chain has 3 steps:

- The provider defines a “wholesale price”
- The provider, publishing his products on the DOME shared catalogue, entitles a DISTRIBUTOR (the DOME operator) to make it available in a centralised catalogue. This DISTRIBUTOR role needs to have some revenue to cover his own expenses.
- The DISTRIBUTOR shares his centralised catalogue to a set of RESELLERS (the federated marketplaces, including the DOME portal) that will take care of the responsibility to show that offering to the consumers (re-selling the catalogue products). This reselling activity is covered with additional revenue.

The sum of wholesale + %distrib. + %resel. = final price



Just to provide an example:

Wholesale price = 80€

Distrib. Fee = 10%

Resel. Fee = 10%

Final price = 100€

(100€ - 20% = 80€)

According to the above description, the DOME operator will be the default distributor supporting all the transactions, the reselling marketplace will be the reseller, and the product owner the provider.

This includes a contractual relationship between the distributor (DOME) and the possible resellers (federated marketplaces).

5.4.6.4 Relationship with the billing process

Having a clear visibility of the wholesale price and the final price for every component, it's quite easy for the billing engine to calculate the customer bill, the distributor fee, the reseller fee, and the provider incomes. The integration of the revenue sharing computation within the billing process has been considered in the previous section.

5.4.6.5 Relationship with the invoicing process

From a formal point of view, due the existence of a "reselling contract" between provider and reseller, the process needs to address 3 different invoicing steps:

- An invoice from the reseller to the customer
- An invoice from DOME to the reseller (federated marketplaces) if the case
- An invoice from the provider to DOME (as distributor) if the case

Those different invoices will have the same cost items but different cost/price values.

The different invoices may have a different tax management too, due the possible different regulations existing in the mother country of the different actors.

5.4.7 Invoicing

5.4.7.1 Overview

In the complex and interconnected world of business transactions, invoices are pivotal as time-stamped commercial documents that record and itemise transactions between buyers and sellers. These formal documents encapsulate crucial details about the deal, including purchase terms and available payment methods.

An invoice must be clearly identified as such and includes a unique invoice number that serves as a reference for internal and external purposes. It also provides essential contact information for the seller or service provider to address any billing-related concerns. The invoice goes beyond merely stating the amount owed, as it outlines the unit costs, total quantities purchased, freight charges, handling fees, shipping costs, and relevant tax charges. This comprehensive information ensures transparency and accountability in financial transactions.



As at this stage of the project, we are unable to include all the technical and legal aspects of the DOME invoice process, this current version of the document provides a general overview of what an invoice is and how it will be used in the DOME marketplace. A more detailed explanation, including input from legal departments, will be included in the next version of this Deliverable (D3.4, M18).

5.4.7.2 What is an Invoice in the DOME Marketplace?

In the context of the DOME marketplace, an invoice is a detailed commercial document that shows the specifics of a transaction between a buyer and seller, including a time stamp. It serves as an official record of the goods or services purchased by the customer and it is essential for connecting the billing and payment processes.

As part of the opportunities presented by the digital age, e-invoicing has been chosen as standard within the DOME ecosystem. E-invoicing plays a key role in ensuring smooth and efficient transactions while upholding the integrity of contractual agreements. The digital nature of e-invoicing also significantly boosts audibility, simplifying the process of tracking and reviewing transactions for compliance and accountability. By reducing paper usage, e-invoicing aligns with environmentally conscious practices, making it a sustainable choice for businesses. Additionally, the capacity for efficient data collection facilitates comprehensive business intelligence, empowering organisations to gain valuable insights from their invoicing processes.

Invoices within the DOME marketplace will include essential components like:

- Invoice Number;
- Contact Information;
- Payment Terms;
- Goods or Services Description;
- VAT Information.

Incorporating these components into invoices within the DOME marketplace is essential for streamlining financial transactions, enhancing communication, and maintaining compliance with legal and regulatory obligations. These components collectively contribute to the smooth functioning of the marketplace and foster a secure and efficient ecosystem for buyers and sellers alike.

As already mentioned at the beginning of this chapter, we delve deeper into the invoicing process in the subsequent version of this document, we will explore the involvement of legal experts to provide a comprehensive understanding of the invoicing procedures within the DOME marketplace.

5.4.8 Payment

This section details the payment subsystem, including payment gateway connection handling, payment instrument saving and managing clearing and settlement (in case of the centralised payment gateway account management version), including the handling of bank transfers as well.

The payment module does not process or store sensitive bank card data, so we do not need PCI DSS compliance or certification. All this data will be processed and stored by the (PCI DSS certified) payment gateways and the payment module will handle only the so-called



security tokens provided by the payment gateways. Unfortunately, this has some drawbacks, like for example we do not know the expiry date of a bank card in advance.

In case of the centralised concepts (both for the bank card based payments and for the bank transfer based payments), the legal entity who will operate the payment service should have a centralised bank account, and for a limited time period this bank account will possess the money paid by the customer to the product provider. There is a need for a legal framework for this concept, and the detailed solution is under discussion with the regulatory authorities.

5.4.8.1 Main technical overview

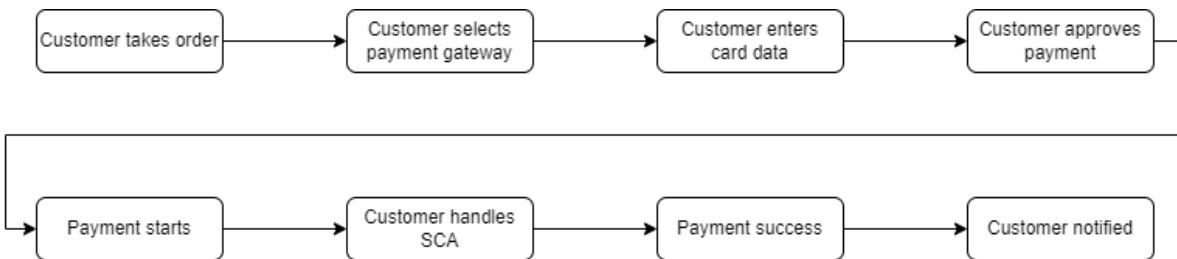
The system contains a

- web application,
- backend services,
- API which is used by TM Forum APIs / Product provider / Web application (payment frontend)

5.4.8.2 Main scenarios

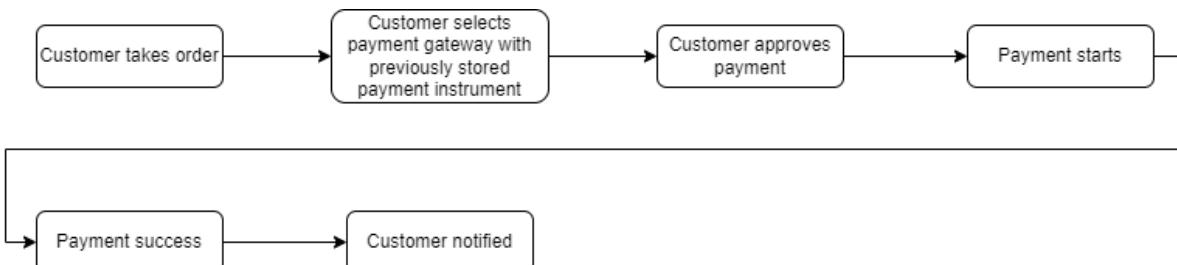
5.4.8.2.1 Payment with new card, with SCA (Strong Customer Authentication)

In this scenario, the customer should enter the bank card data (bank card number, expiry date, CVC code, and sometimes the name written on the card as well) during the payment process. This scenario includes the SCA process (Strong Customer Authentication) as well, which means an additional authentication initiated from the card issuer bank. The customers will be asked to verify their identity with two factors during the payment process.



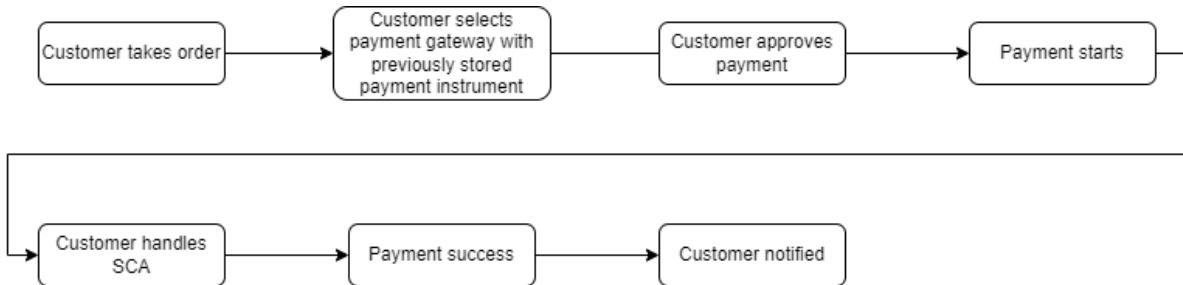
5.4.8.2.2 Payment with existing card, without SCA

In this scenario, the customer can reuse a previously stored bank card data for the actual payment process. This scenario shows a process without SCA. This means that this actual payment transaction was flagged as an Out of Scope or Exempt Transaction by the issuer bank.



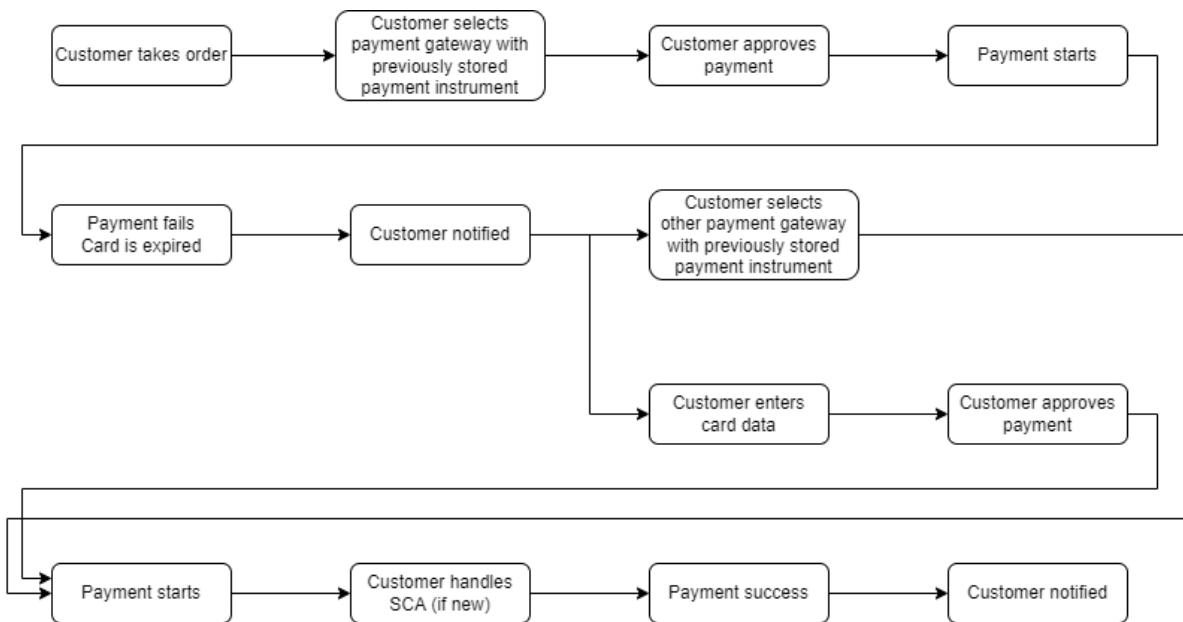
5.4.8.2.3 Payment with existing card, with SCA

In this scenario, the customer can reuse a previously stored bank card data for the actual payment process. This scenario shows a process with SCA.



5.4.8.2.4 Bank card data update during payment because of expiration

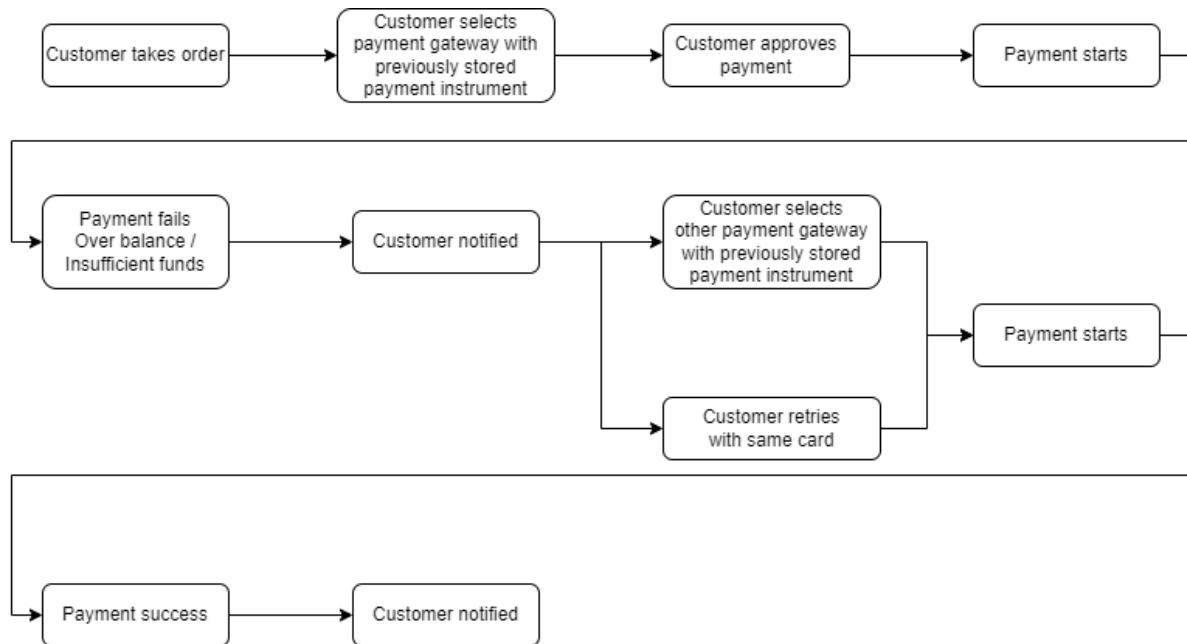
In this scenario, the customer would like to reuse a previously stored bank card data for the actual payment process, but the bank card is already expired. So the customer should choose another bank card, or update the card data. This scenario shows a process with an optional SCA.



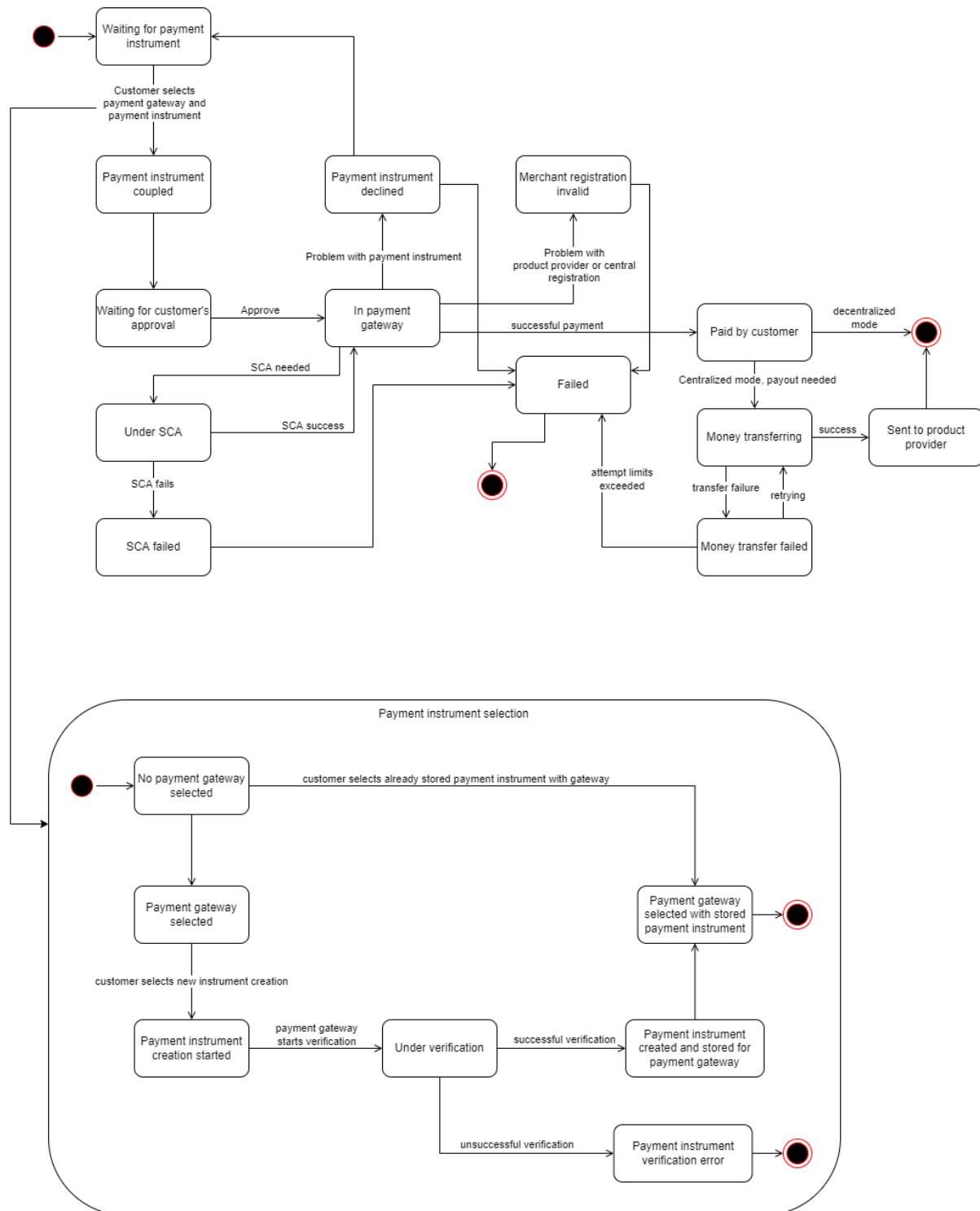
5.4.8.2.5 Alternative bank card selection during payment because of insufficient funds, without SCA

In this scenario, the customer would like to reuse a previously stored bank card data for the actual payment process, but there is not enough funds on the bank account connected to the given bank card (or some limits have been reached). So the customer should choose another bank card, update the limits of the given card or make more funds available. This scenario shows a process without SCA.





5.4.8.3 State diagram



5.4.8.4 Main concepts

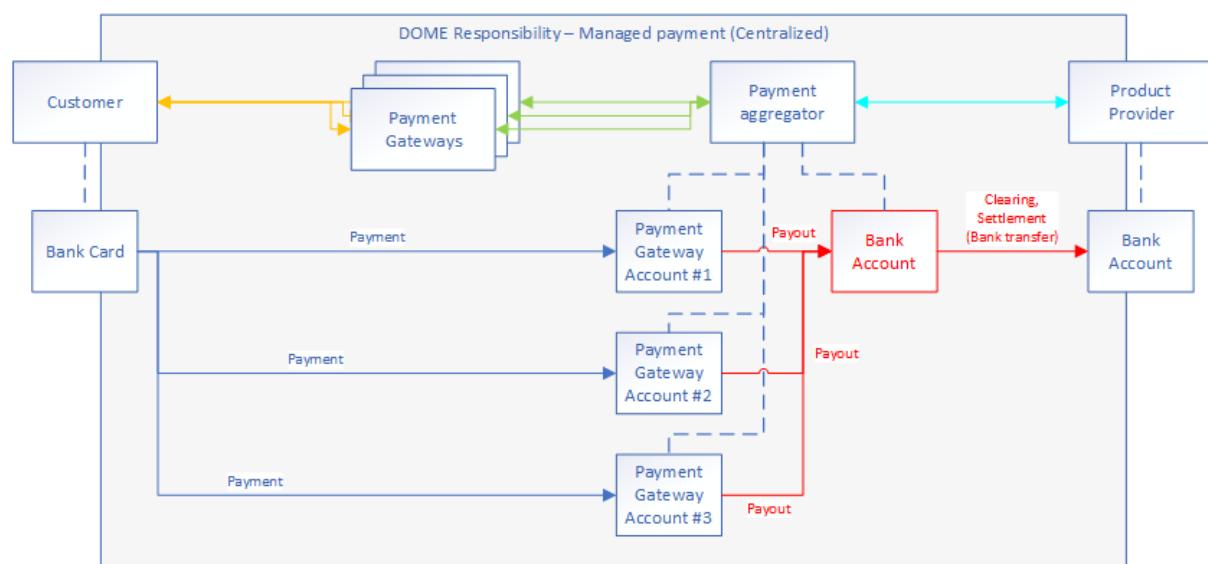
This subsystem is responsible for receiving the payment transaction related information:

- amount
- currency
- payment method
- country of location
- product provider information (e.g. gateway registrations)
- customer payment related information (e.g. saved payment instrument tokens)

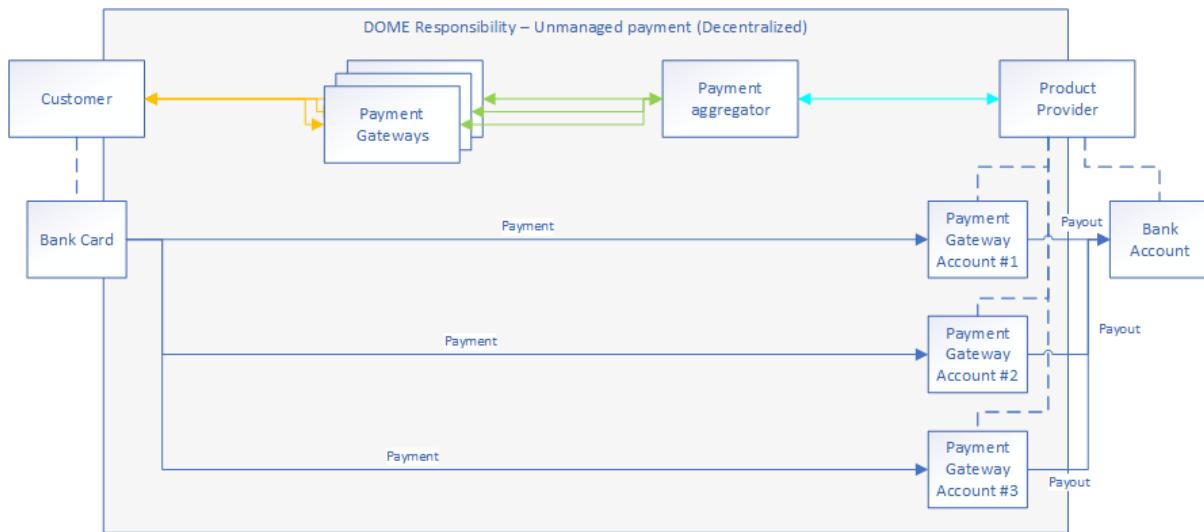
Then starting a transaction with these parameters - transformed and specialised - to the customer's chosen payment gateway.

There are two main concepts planned, the main difference is the connection to the payment gateways.

In the centralised concept (detailed in section 5.4.8.5) the system is responsible for registering and maintaining the connection to each payment gateways.



In the decentralised concept (detailed in section 5.4.8.6) each product provider must register itself to the payment gateways.



In each concept the system will support each payment gateway, and makes it easy to connect a new one.

These two approaches have their own set of advantages and disadvantages, making them suitable for different business models and objectives. Let's summarise the key differences and merits of these two payment aggregation solutions.

In the managed payment solution model (where we will use centralised payment gateway accounts), the payment aggregator takes on the responsibility of establishing and managing payment gateway accounts on behalf of product providers.

The advantages of the managed payment solution:

- Financial and legal compliance: one of the standout strengths of the managed payment solution is its ability to handle complex financial and legal aspects efficiently, the payment aggregator is equipped to navigate the regulatory landscape and ensure compliance with industry standards, reducing the burden on product providers
- Ease of onboarding: product providers can seamlessly integrate with any payment gateway without the hassle of individually creating and maintaining accounts with multiple providers, this streamlined onboarding process saves time and resources
- Flexibility for buyers: the managed payment solution offers full flexibility for buyers, allowing them to choose their preferred payment gateway, this flexibility enhances the overall customer experience and can lead to higher conversion rates
- Lower transaction costs: centralised management enables the payment aggregator to negotiate better terms with payment gateways, resulting in lower transaction costs for product providers, this cost reduction can have a significant impact on a business's bottom line
- Revenue sharing: the managed payment solution can facilitate revenue sharing through a transaction fee collection based on a DOME business model

The disadvantages of the managed payment solution:

- Financial and legal aspects: payment licence and strict company procedures and workflows will be required

In the unmanaged payment solution model, in contrast, places the responsibility of creating and managing payment gateway accounts squarely on the shoulders of the product providers. This approach offers a different set of advantages and disadvantages.

The advantages of the unmanaged payment solution:

- Reduced responsibility: DOME has significantly less responsibility in this model, as the product providers require to create and maintain their own payment gateway accounts, this approach is particularly appealing to such product providers that prefer to maintain full control over their payment processes
- Direct revenue: product providers can receive payments directly from the payment gateways, eliminating intermediaries and potentially speeding up access to funds

The disadvantages of the unmanaged payment solution:

- Account creation overhead: one of the most significant challenges of the unmanaged payment solution is the need for product providers to create accounts with multiple payment gateways, this can be a time-consuming and resource-intensive process, especially when dealing with a large number of gateways
- Limited buyer flexibility: the unmanaged model may limit the flexibility for buyers, as they may not have the option to choose their preferred payment gateway, if a given product provider lacks the support for them, this lack of choice could lead to a less optimised customer experience
- Higher transaction costs: without the negotiating power of a centralised aggregator, product providers may face higher transaction costs from individual payment gateways, over time, these costs can add up and impact profitability
- Complex revenue sharing: establishing revenue-sharing process in the unmanaged model can be more challenging, requiring three-party payments or depending on separate payment processes from the product providers side

5.4.8.5 Centralised payment gateway accounts

In the centralised payment gateway concept, this subsystem registers to each supported payment gateway. It maintains the registration, updates tokens etc. what is required for the specific payment gateway. Theoretically every transaction can be executed with this concept, because there is no limitation from the product owner side.

If a transaction begins, a customer is loaded from TM Forum API, with its transaction history and previously saved payment instrument tokens. After selecting a gateway, the transaction will pass to the gateway, and after success, this subsystem will receive the product owner's money, which will be transferred to its bank account.

5.4.8.5.1 Payout

The amount paid by the customer first arrives at the payment gateway account. There is a need for a payout process to the central bank account after that. The timing will depend on the payout schedule of the payment gateway and the payout speed from the payment gateway account to the central bank account.

The payment module will manage the bank transfer from the central bank account to the product provider bank account, as the last step of the payout process. Based on the payment request parameters, there can be multiple beneficiaries for these bank transfers, in case of product bundles.

This could be a fully automated process, or a supervised semi-automated process, depending on the configuration and the requirements of the operational team of DOME.



The revenue sharing, or the transaction fee collection can be an integrated part of this bank transfer process as well.

5.4.8.6 Decentralised payment gateway accounts

In this concept this subsystem receives the product provider's access tokens in a secure way to create a transaction to the gateway. The product provider needs to maintain the connection and registration to each gateway he wants to use.

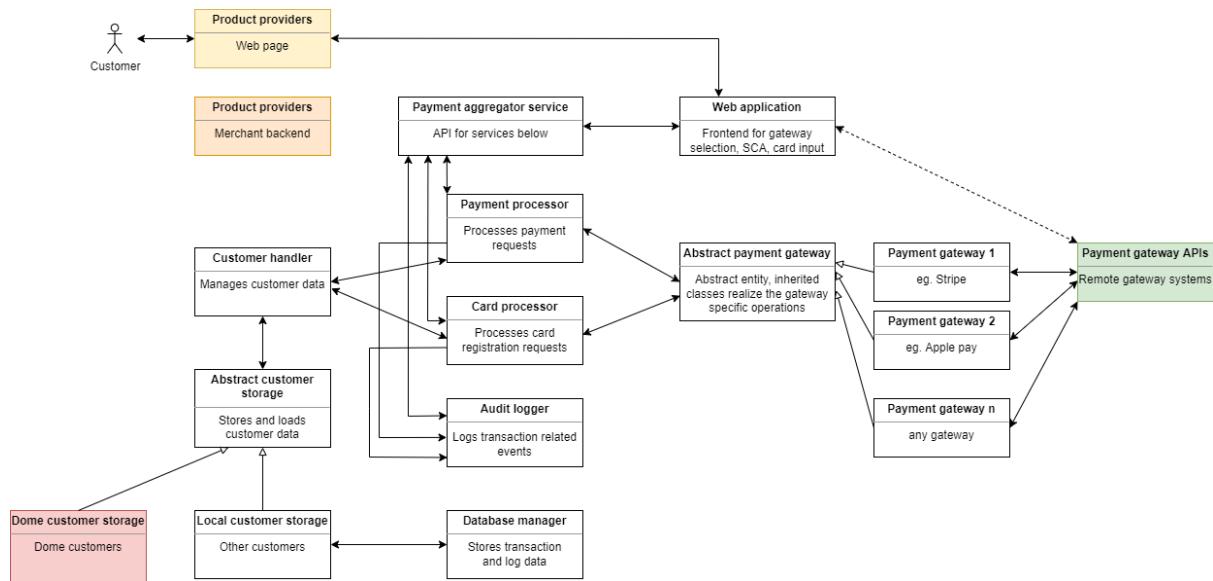
There is no need for payout in this version, but collecting the transaction fee (aka revenue sharing) during the payment process is not possible either, it should be managed outside of the payment module.

5.4.8.7 Payment with bank transfer

The initiation of bank transfers is outside the scope of the payment module. But in the centralised concept we need to manage the incoming bank transfers, and pairing the transfers to the unpaid invoices. Also we need to manage the payout similar to the decentralised payment gateway accounts version.

5.4.8.8 Interactive payments

Interactive flows are requiring the card holders active cooperation during the process. This can mean handling SCA requests from the banks during the transaction or first time checking the payment instrument. The web application will redirect the customer or display gateway specific UI for these events.

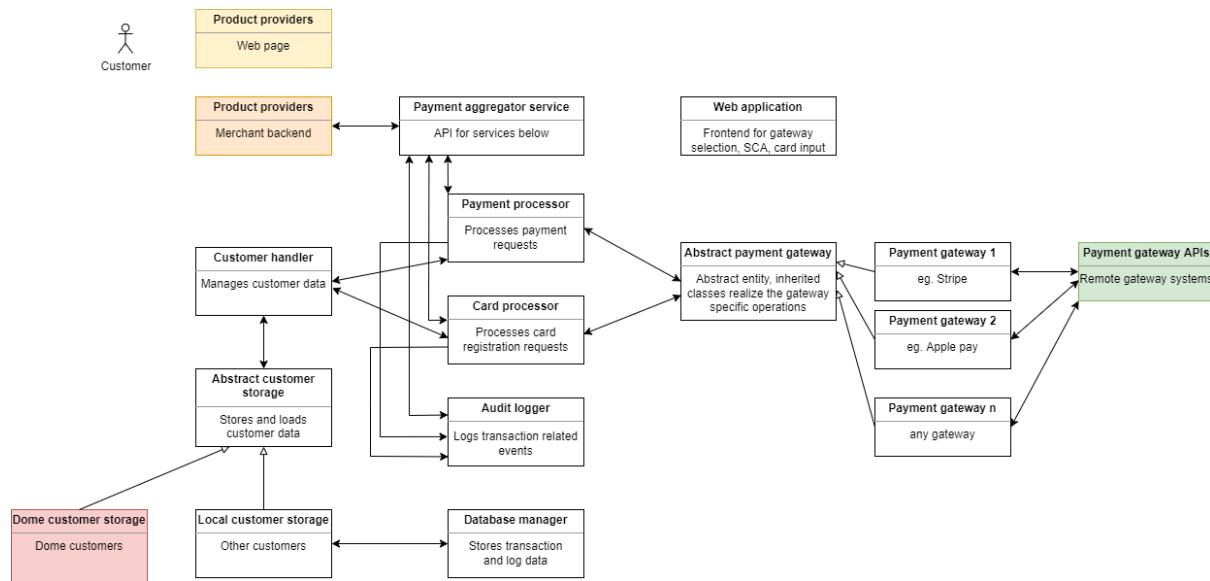


5.4.8.9 Recurring / non interactive payments

Recurring payments will require no UI, but the saved payment instrument's token coupled to the chosen payment gateway. This token was saved intentionally during a previous payment process.

So in normal operation, this kind of process does not require any interaction from the customer. But in case of some problems (like an expired bank card), we should notify the customer and ask for some tasks to complete (like update bank card information). The product

providers should be informed via notification messages as well in case of any failure during the non interactive payment processes (from the TM Forum APIs). These notifications should indicate if the failure is just temporary, or final. The exact process flow depends on the product provider settings (like grace period settings, for example).



5.4.8.10 Backend services

5.4.8.10.1 Payment processor

Payment processor is responsible for receiving and validating payment data from API, and managing payment status switches. It also logs any event related to the payment. It initiates the transaction to the chosen payment gateway provider.

5.4.8.10.2 Card processor

Card processor handles payment instrument token saving, and loading the specific instrument token from the customer storage.

5.4.8.10.3 Audit logger

Logs events with timestamps.

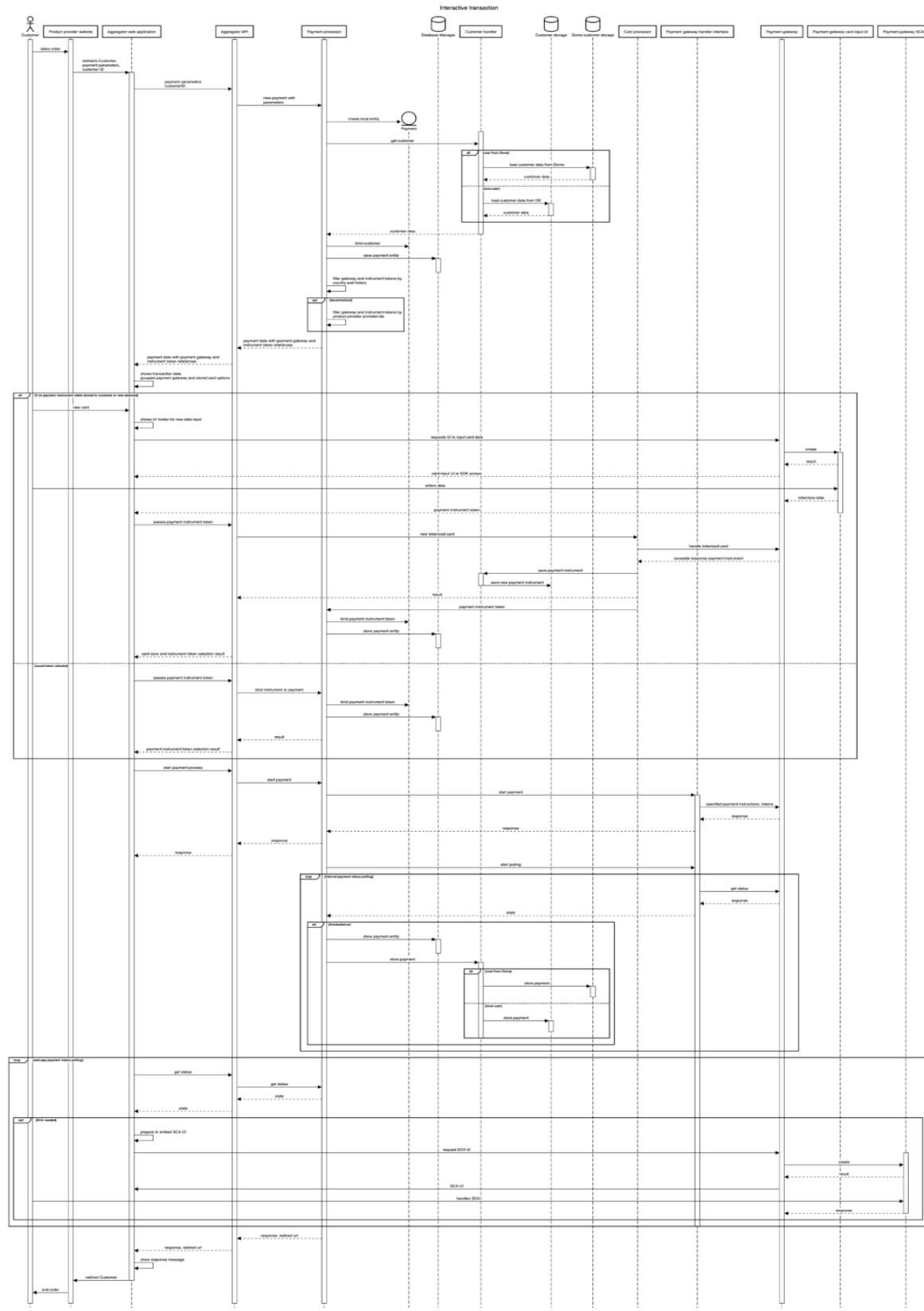
5.4.8.10.4 Customer handler

Responsible for requesting customer data from TM Forum API or local database. Attaches payment instrument tokens to customers.

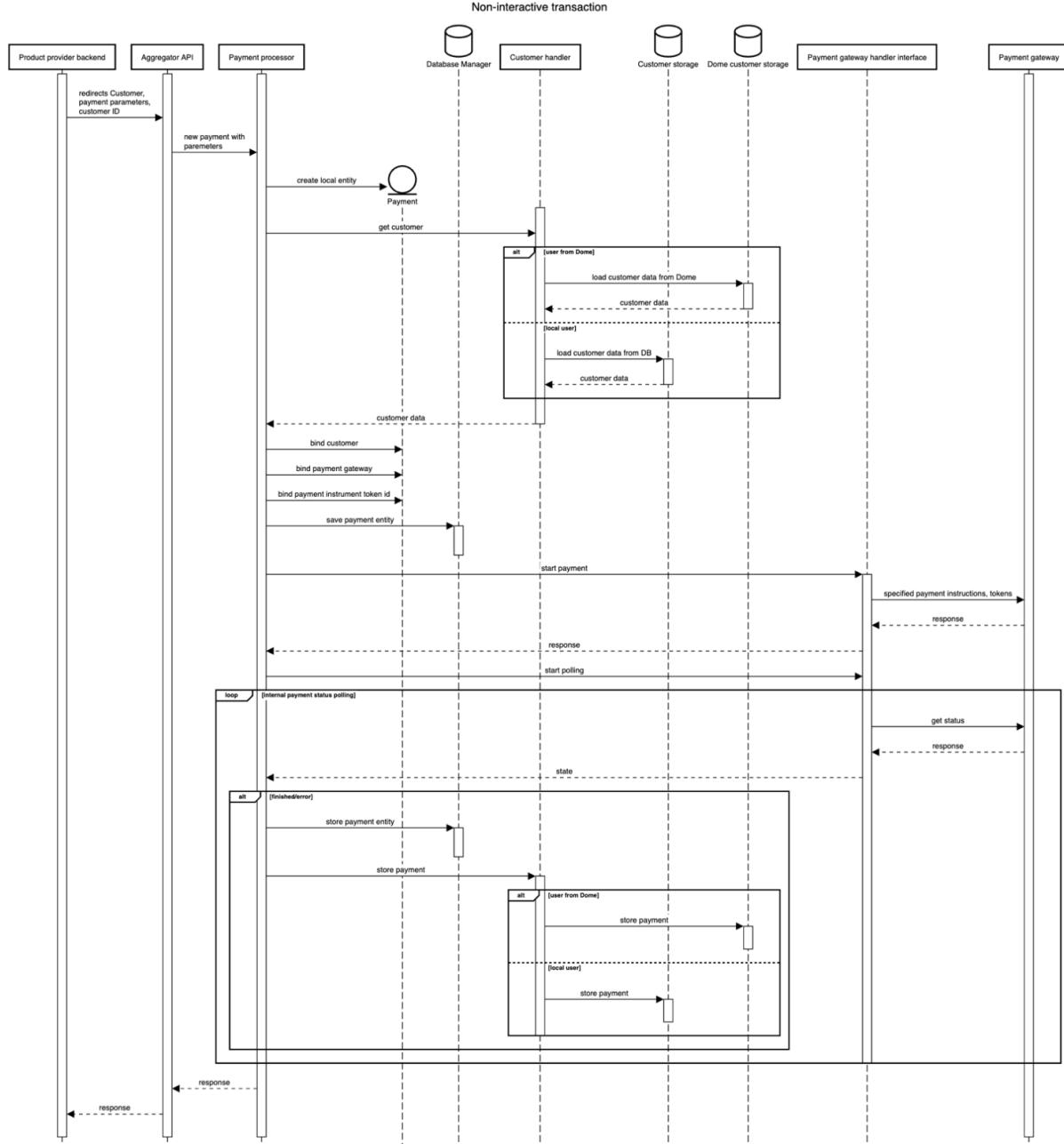
5.4.8.11 Example payment flows

5.4.8.11.1 Interactive flow





5.4.8.11.2 Non-interactive flow



5.5 Indexing and Search

The Indexing and Search component provides basic and advanced features to search, index, analyse and correlate information to be retrieved, with the goal of connecting consumers with relevant and coherent services as quickly as possible. In its most basic functionalities, the DOME central Marketplace and its product catalogue should allow customers to:

- launch product searches on the marketplace portal;



- leveraging category filters (based on domain-oriented taxonomies);
- easily accessible service description and service offering pages.

Also, through the exploitation of advanced search/browsing algorithms, it is possible to design and implement more advanced features with the goal of connecting customers with relevant products in the least possible time. For example, a search algorithm which would match customer search queries with keywords from relevant product listings; even more advanced search functions may leverage additional information (such as product ratings or click-through rates) to prioritise/rank the results of search queries and improve the customer experience.

The approach applied to design the architecture of the Indexing and Search component started from the identification of all the possible requirements. Generally, the practice of search information (and related retrieval) is a process involving several elements:

- a query with information about what the user is trying to retrieve;
- an interface for the query composition and the results visualisation;
- a set of filters that reduce the amount of information, focusing the results on what is useful for the user;
- a storage area, where the information is persistent and can be retrieved.
- a set of results useful for the user.

Specifically for the context of DOME, it is necessary to consider:

- a process that manipulates queries and/or pre-process data in order to add semantic-retrieval capabilities to the system;
- a process implementing the logic needed to index anything that needs to be searched.

In addition, capabilities like sorting, filtering and ranking are very important in an environment working with a huge amount of data. For this reason, the design process must take into account all aspects. Starting from these considerations, the Indexing and Search component must implement several features needed for the search and retrieval task among which:

- search by keywords (allow customers to search and filter published offers using different parameters, including text search, catalogues, categories and offering params such as type, owner, characteristics or status);
- search by Natural Language request (allow customers to search and filter published offerings using requests in Natural Language).

All these functionalities must be exposed as REST services. The following picture describes the proposed solution.



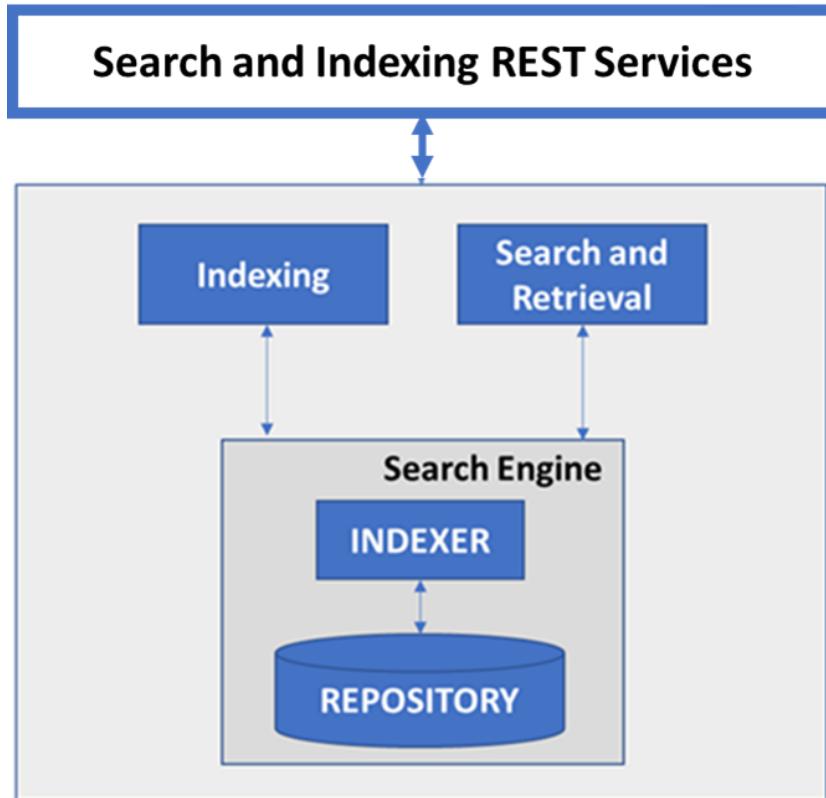


Figure 60 - Search and Indexing: architecture

The component has specialised parts implementing groups of functionalities. The API layer exposes in the form of REST services all functionalities implemented by the component. The Search and Retrieval section implements high-level APIs focused on management of queries submitted by the DOME user; functionalities for advanced search operations and results creation and presentation are also exposed. The Indexing section is specialised in strategies to index Offerings, public items and catalogues in terms of their contents. The Indexing and the Search and Retrieval subsystems will be detailed in next paragraphs.

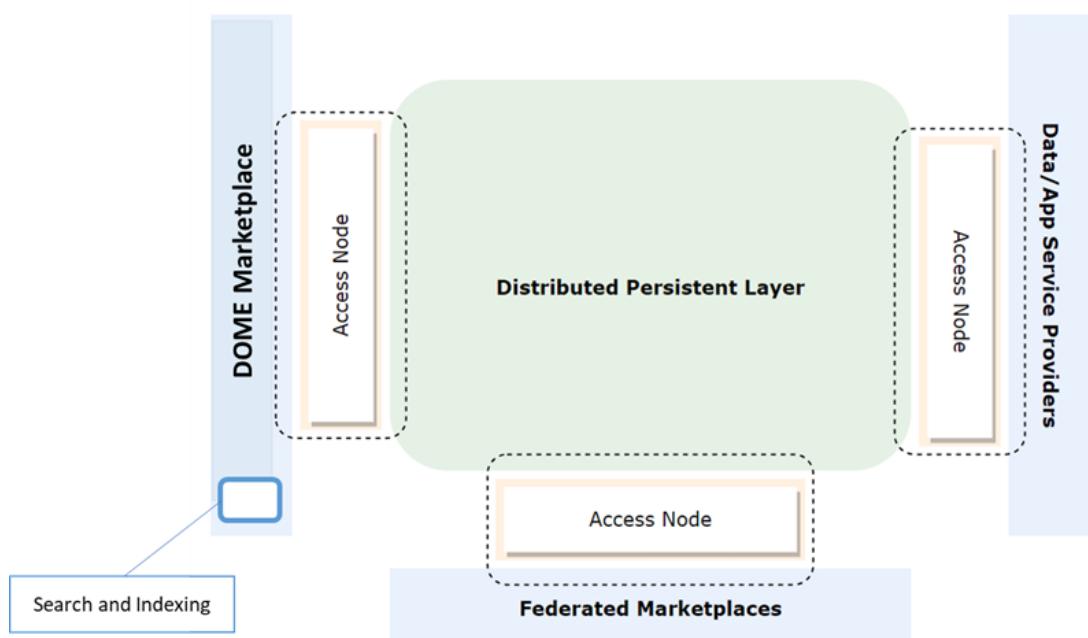
The Indexing and Search component requires a document database able to store unstructured data such as profiles, catalogues, and large documents. The choice has been ElasticSearch, a resilient and scalable document database. Responses are very fast and it integrates a customizable indexer that reduces the effort needed to solve the indexing problem characterising the search and retrieval operations. ElasticSearch can be integrated through REST API or Transport Client in order to ensure robust communications. As said, the component communicates with the other parts of the system through set of APIs exposed as REST services: about the technological choices related to the implementation of the component, Spring Boot2 is under analysis as promising technological choice for the following features:

- It allows the creation of single package components (in form of Jar including the web server);
- There is an extension for the automatic health check;

- It allows the implementation of robust REST services in order to expose the API;
- Through Spring Cloud it is possible to easily integrate Kafka, which is a standard solution for message exchange based on publish-subscribe pattern.

Spring framework also allows the communications with service registries (like Eureka) in order to implement an architecture compliant with the micro-services philosophy.

Concerning the architectural approach and positioning in relation to the DOME Marketplace, the proposed solution considers the Indexing and Search component as a subsystem of the DOME central Marketplace, behind the same access node. Thus, the communication channel is Blockchain: when an event is published, Marketplace and Index and Search receive the same notification. In this context, the Indexing and Search services represent a value added for the DOME Marketplace. The detailed description of the main aspects for this solution is planned for the next version of deliverable.



Search Engine will rely on the DOME Central Marketplace Catalogue. Offerings and public items stored into the central marketplace will be indexed for search and retrieval purposes. Specific mechanisms to index and keep up to date catalogues and related items of the available federated marketplaces will be analysed and implemented following a decoupling approach in order to implement a unique interface independently from the nature of the services offered and the specificity of the marketplace.

5.5.1 Indexing subsystem

Indexing is a process of extracting index terms from a large pool of documents collection and organising them using indexing structure to facilitate fast and accurate information retrieval. The purpose of storing an index is to optimise speed and performance in finding relevant

documents for a search query. Without an index, the search engine would scan every document in the corpus, which would require considerable time and computing power.

Within the DOME context, two main functions have been established:

- indexing of a new item;
- indexing of a catalogue.

These features will be detailed in the scenarios described below.

5.5.1.1 Indexing a new Item

The following sequence diagram summarises the “indexing of a new item” workflow specific for the context of DOME:

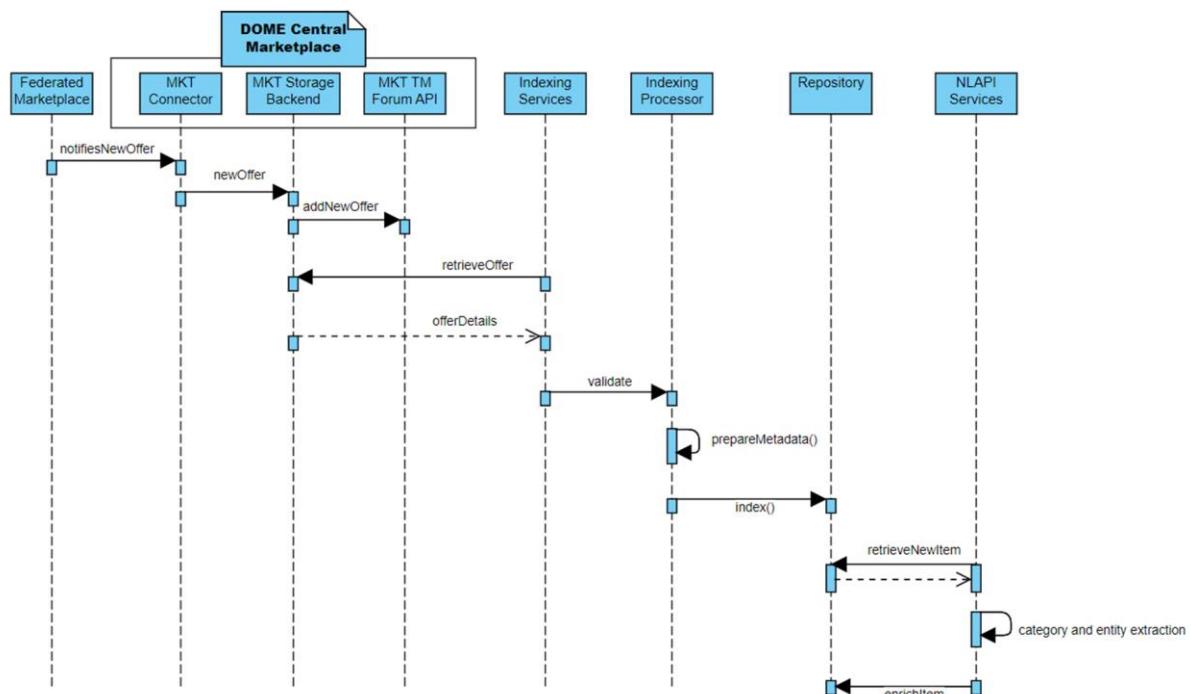


Figure 61 - Indexing a new item

The Indexing process is started/triggered every time a new item (Offerings or public Items) is created and saved in the DOME central Marketplace.

Once the new offer is validated and stored by the MKT Storage Backend, all the details related to the offer are retrieved by services that have in charge the indexing process. An automatic process validates information just retrieved and prepares the metadata schema to be processed by the indexing operation.

At this point, the indexing process is finalised through the Repository: the schema is stored and available semantic services could enrich the metadata through the category and entity extraction process.



5.5.1.2 Indexing a Catalogue

The indexing of a catalogue could be intended as an operation that allows to index a catalogue related to a marketplace. The following sequence diagram is specific for the DOME Marketplace shared catalogue:

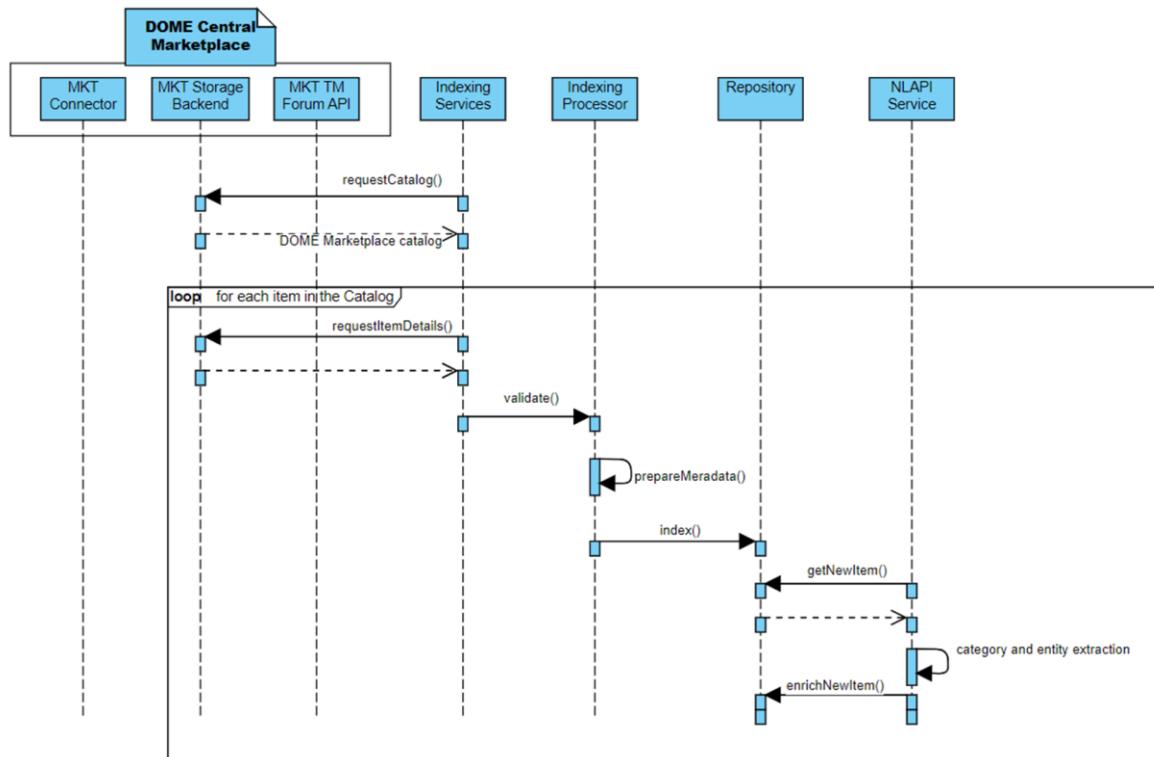


Figure 62 - Indexing a catalogue

The Indexing process starts with the request of the catalogue to the specific marketplace: for each item in the catalogue (Offerings or public Items) all the related details are retrieved by the services that have in charge the indexing process. Thus, an automatic process validates information just retrieved and prepares the metadata schema to be processed by the indexer operation.

At this point, the indexing process is finalised through the Repository: the schema is stored and available semantic services could enrich the metadata through the category and entity extraction process.

5.5.2 Search subsystem

The purpose of the search and retrieval functionalities is to implement a process that helps DOME users to retrieve useful offerings. The most important steps of a search process include:

- A user submits a query to the system through a graphical interface;



- The component, implementing the search and retrieval functionalities, processes the request and retrieves search results;
- The search results are presented through a graphical interface (e.g. in form of card list);
- The user selects a result that can be entities or resources for other operations (e.g. open details);

The following sequence diagram summarises the search workflow specific for the context of DOME:

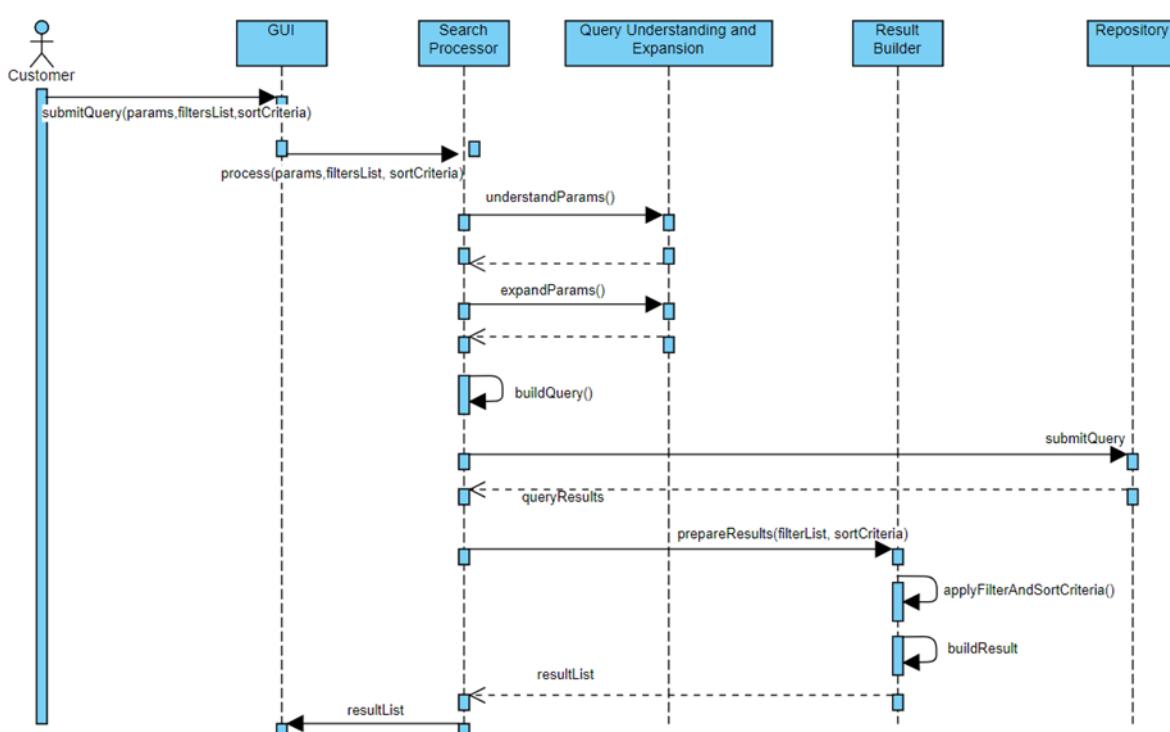
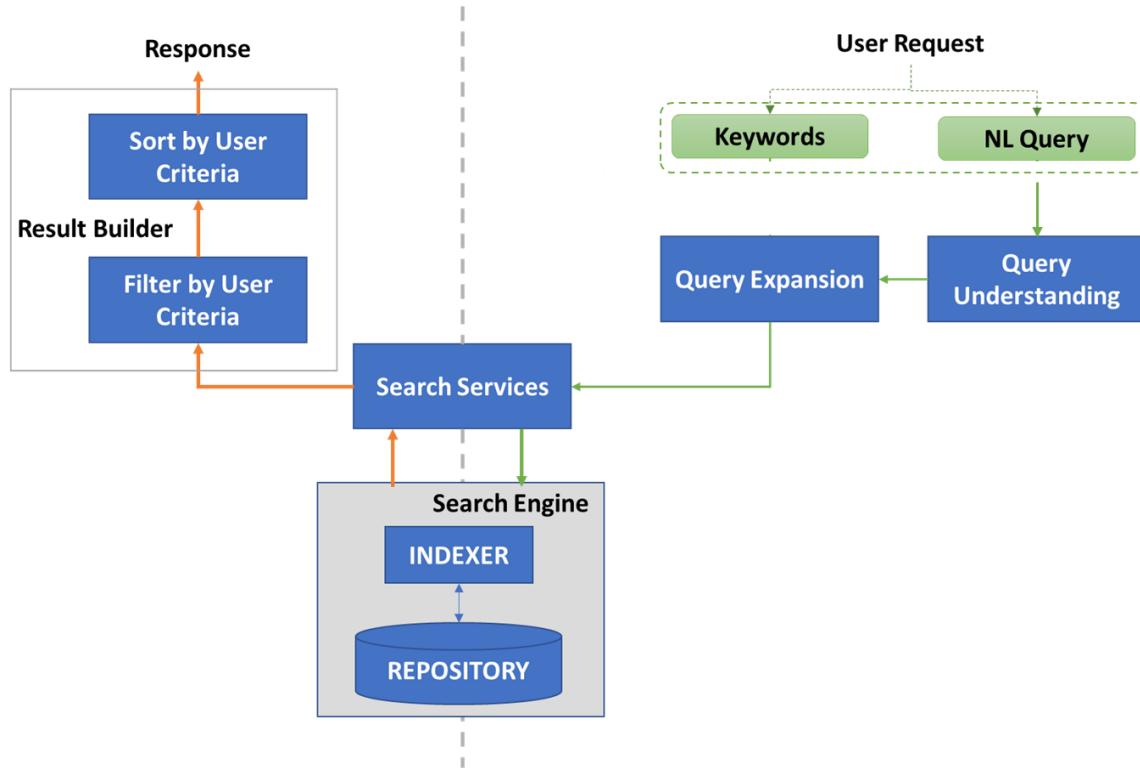


Figure 63 - Search workflow

A user submits a request to the system through a graphical interface: a request could be composed of keywords to be searched, a list of filters and a list of criteria to be used during results sorting. Furthermore, the user can submit a request in Natural Language. When the request is completed, the query is submitted.

- Assuming that users do not always formulate search queries using the best terms, the user query is pre-processed using available semantic services by extracting semantic meaning from the searcher's keywords/NL query (Query Understanding) in order to prepare it for a semantic expansion (Query Expansion). During the query expansion stage, the user's input is expanded with additional terms (such synonyms of words, as well as semantically related words) with the aim of improving performance of retrieval operations by the production of better search results.
- At this point of the workflow, the query is expanded but can require a high computational cost to be executed. For this reason, an automatic process will optimise the query in order to speed-up the search process (Search Processor). The Search processor decomposes the complex query in multiple queries resolved through the Repository in order to retrieve results.

- All results coming from queries are merged in one single results list. Also, in order to refine the results, the search processor submits the list of results to a specialised result builder producing a more compact and precise list of results.
- The results list is the input for a first filtering process (filter by User Criteria) that takes care of users filtering options. In this way, it is possible to obtain a filtered list of results that will be finally ordered by user criteria (sorted by User Criteria).



5.5.3 NLAPI Services

In order to support the indexing and the searching activities, a set of functions based on NLP (Natural Language Processing) and NLU (Natural Language Understanding) have been provided [7].

Deep linguistic intelligence makes it easy to quickly analyse text for key elements, relationships, classifications and more, making unstructured data structured. The referred functionalities are the Document Analysis and Document Classification functions.

⁷ Understanding NLP vs NLU vs NLG,
<https://www.expert.ai/blog/dont-mistake-nlu-for-nlp-here-why/#:~:text=While%20both%20NLP%20and%20NLU,language%2C%20NLU%20provides%20language%20comprehension>

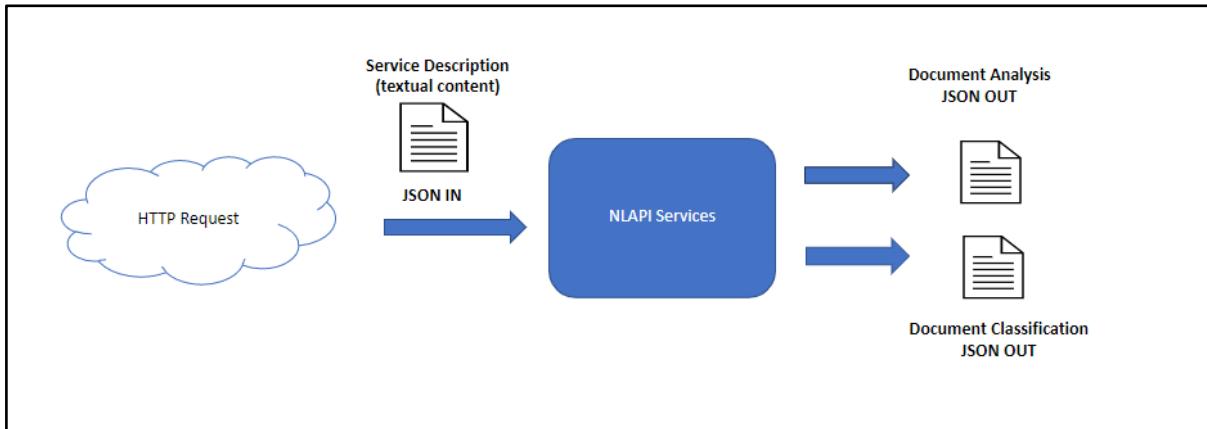


Figure 64 - NLAPI Services Architecture

5.5.3.1 Document Analysis

The Document Analysis function allows to disambiguate text and extract key elements, entities, topics, sentiment and more, making sense to unstructured data. The deep linguistic analysis combines text subdivision, part-of-speech tagging, morphological analysis, lemmatization, syntactic analysis and semantic analysis.

Given a textual content, submit it to a document parsing function based on NLP and NLU, it means carrying out a disambiguation process that allows you to have:

- Detailed syntax identifying tokens, part-of-speech tags, phrases, and sentences
- Full dependency tree
- Concept labels pinpointing the precise meaning of each term in the sentence by leveraging context.

Document Analysis Service Description

The json input request must have the structure shown in the figure below:

```
{
  "document": {
    "text": "Your text here."
  }
}
```

Figure 65 - Document Analysis Service JSON IN

The document analysis service processing, for the considered purpose, will carry out what is defined as Full Document Analysis. The Full Document Analysis is structured as follows:

- ***Deep linguistic analysis:***



- Text subdivision: Paragraphs, Sentences, Phrases, Tokens, Atoms detection
 - Part-of-speech tagging: marking up each token with the corresponding Universal POS tag
 - Morphological analysis: lexical and grammatical features of each token
 - Lemmatization: tags tokens and atoms with their corresponding lemmas
 - Syntactic analysis: parsing process that detects the universal dependency relation between each token and the sentence root token or another token
 - Semantic analysis: maps tokens to the Expert.AI internal concepts storage
-
- **Keyphrase extraction:**
 - Relevant topics
 - Main sentences
 - Main phrases
 - Main concepts
 - Main lemmas
 - **Named entity recognition:** which entities—persons, places, organisations, dates, addresses, etc.—are mentioned in a text and the attributes of the entity that can be inferred by semantic analysis.
 - **Relation extraction:** labels concepts expressed in the text with their semantic role.

Main concepts are returned as Expert.AI internal concepts storage "synccons" and enriched through knowledge linking: open data—Wikidata, DBpedia and GeoNames references—are returned. In the case of actual places, geographic coordinates are also provided. Relevant topics are chosen from the Expert.AI internal concepts storage.^[8]

Named entity recognition, relation extraction also perform knowledge linking: Expert.AI internal concepts storage information and open data —Wikidata, DBpedia and GeoNames references — are returned for entities, relation items, text items that express sentiment corresponding to synccons of the Expert.AI internal concepts storage.

The JSON output will have the structure shown in the figure:

⁸ Standard context topics, <https://docs.expert.ai/nlapi/latest/guide/keyphrase-extraction/>



```
{
    "success": Boolean success flag,
    "data": {
        "content": analyzed text,
        "language": language code,
        "version": technology version info,
        "knowledge": [],
        "paragraphs": [],
        "sentences": [],
        "phrases": [],
        "tokens": [],
        "mainSentences": [],
        "mainPhrases": [],
        "mainLemmas": [],
        "mainSyncons": [],
        "topics": [],
        "entities": [],
        "relations": [],
        "sentiment": {}
    }
}
```

Figure 66 - Document Analysis Service JSON IN

5.5.3.2 Document Classification

The Document Classification function allows analysing text to label and identify media topics, emotional traits, geographical references and more, allowing to classify and to categorise texts. It is based on the Expert.AI internal concepts storage along with the IPTC⁹ taxonomy, to allow wider ways to classify and categorise texts. The IPTC Media Topics taxonomy is the global standard for news media.

Process a document by Document Classification based on IPTC taxonomy, it means intercepting, within the processed text, the textual sections that refer to the managed IPTC categories.

Document Classification Service Description

The json in, in the service request, must contain the text to classify; it will have the structure shown in the figure below:

⁹ IPTC Taxonomy <https://iptc.org/standards/media-topics/>



```
{  
  "document": {  
    "text": "Your text here."  
  }  
}
```

Figure 67 - Document Classification Service JSON IN

Submitting a text to the classification process means to determine what the text is about, on the basis of a given taxonomy category.

The provided Categorization Service is based on the IPTC Taxonomy and the json out from it will report indications regarding the success or failure of the classification operation, the precise content of the input text, the language in which the text is written and the array of the intercepted categories, as shown in the following figure:

```
{  
  "success": Boolean success flag,  
  "data": {  
    "content": analyzed text,  
    "language": language code,  
    "version": technology version info,  
    "categories": []  
  }  
}
```

Figure 68 - Document Classification Service JSON OUT

The array categories elements will be json objects of the following type:



```
{
    "frequency": 70.62,
    "hierarchy": [
        "Sport",
        "Competition discipline",
        "Basketball"
    ],
    "id": "20000851",
    "label": "Basketball",
    "namespace": "iptc_en_1.0",
    "positions": [
        {
            "end": 14,
            "start": 0
        },
        {
            "end": 53,
            "start": 35
        },
        {
            "end": 139,
            "start": 136
        }
    ],
    "score": 4005.0,
    "winner": true
}
```

Figure 69 - Document Classification Service JSON OUT Categories

In each category json object there will mainly be the following informations:

- **Namespace**: the name of the software module containing the reference taxonomy
- **id**: category identifying code
- **label**: category description
- **hierarchy**: category membership hierarchy
- **score**: category cumulative score
- **frequency**: the percentage ratio of the category score to the sum of all categories' scores
- **winner**: a Boolean flag set to true if the category was considered particularly relevant
- **positions**: an array containing the positions of the text blocks that contributed to category score.



5.5.3.3 NLAPI Use Example

An example of practical use of the NLAPI functions is described below.

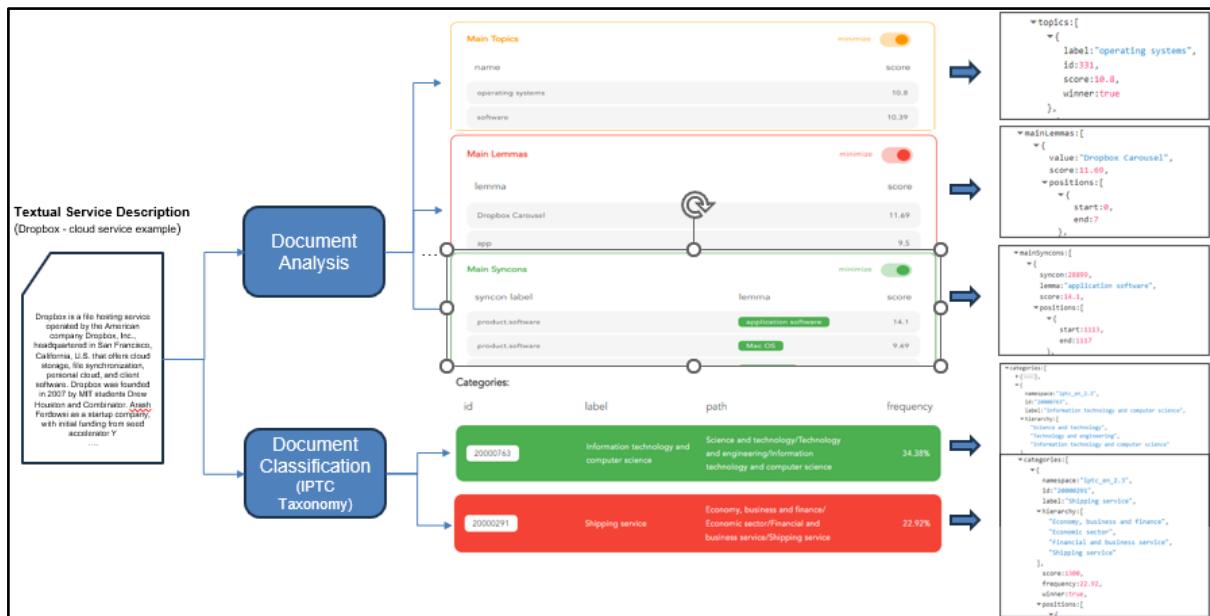


Figure 70 - NLAPI Use Example

Supposing the submission of a file hosting service textual description, in this case Dropbox*, to the NLAPI functions described above. The Document Analysis Function will report, in the json out:

1. main topics, coming from a recognition activity of main arguments dealt with in the document, such as "software", "computer science", "commerce", as shown in the figure below:

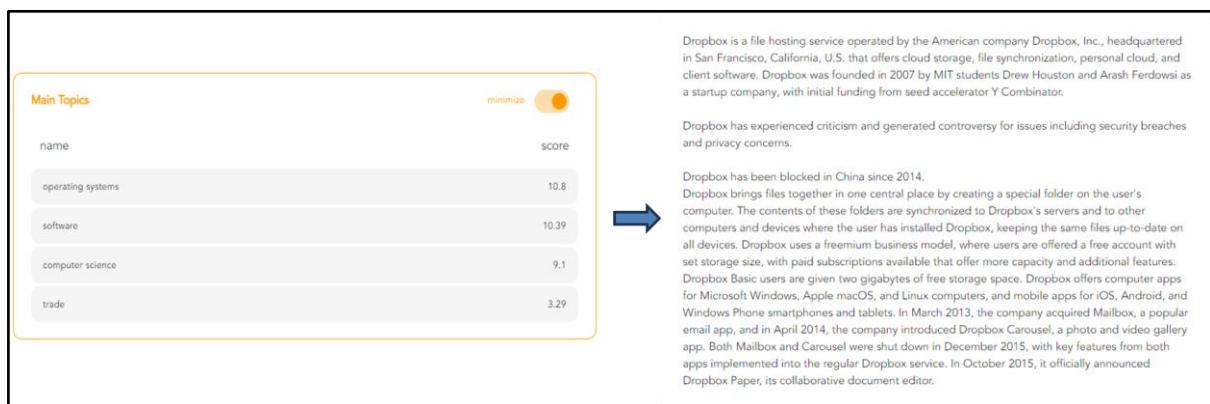


Figure 5.5.3.3.2 - Dropbox Main Topics

2. main lemmas, that represent the key concepts mentioned within the document, such as “Dropbox Carousel”, “app”, “software”, as follow:



Figure 71 - Dropbox Main Lemmas

3. main sentences, that is the sentences that bring the principal informative charge of the entire document; they are selected by exploiting sophisticated linguistic algorithms:

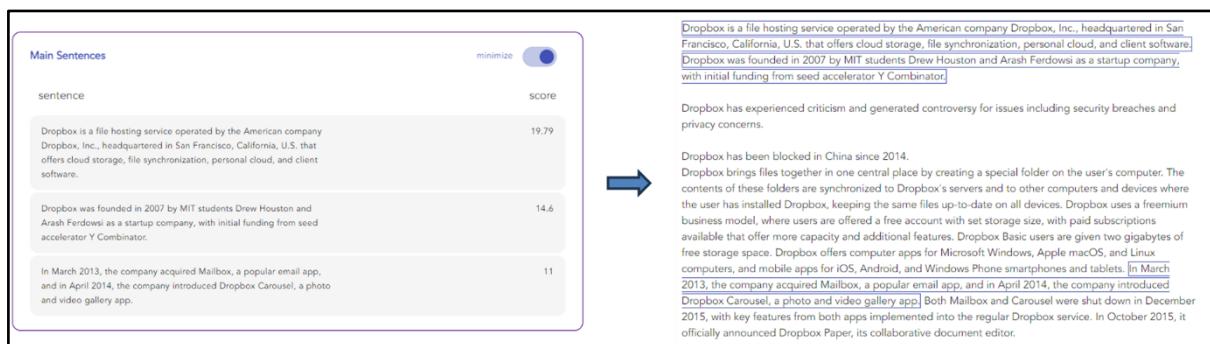


Figure 72 - Dropbox Main Sentences

4. main phrases, that are mainly composed lemmas that describe advanced concepts:



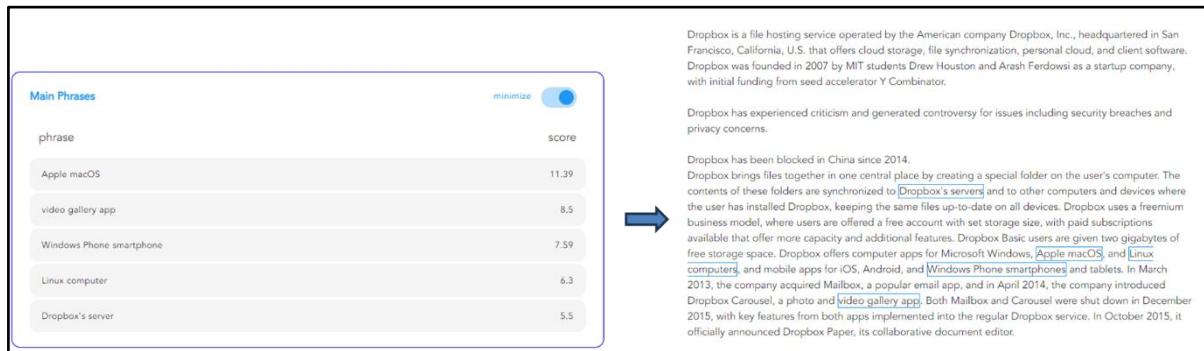


Figure 73 - Dropbox Main Phrases

5. labels like “product.software”, “artifact.instrument” and so on, that come by exploiting the linguistic semantic network at the basis of the NLP technology:

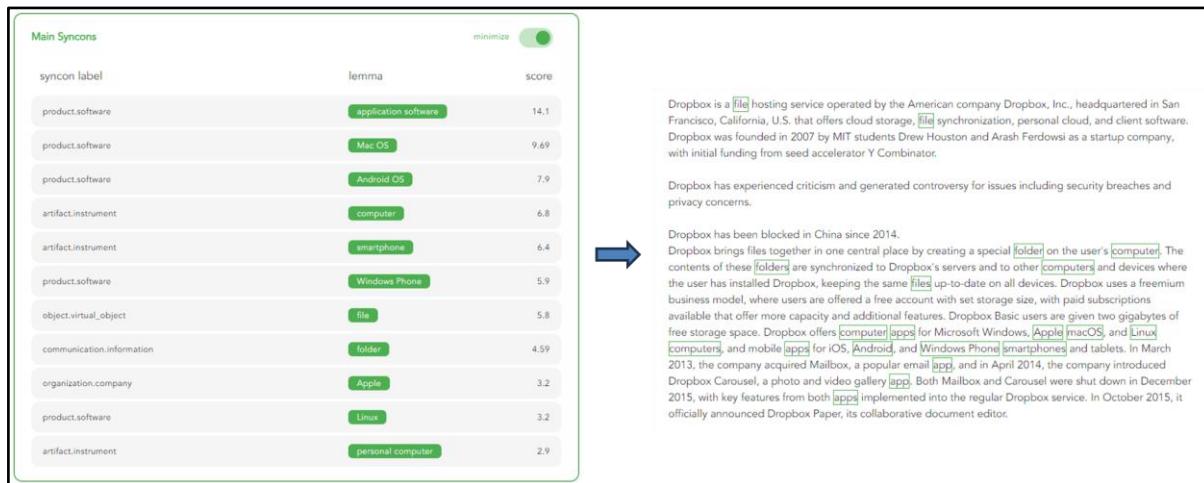


Figure 74 - Dropbox Main Synccons

6. entities such as “Product”, “Activity/Company”, “Organization” and other property values expected from the full document parsing function:



element	type	linked data
Arash Ferdowsi	Person	
Drew Houston	Person	
Dropbox Paper	Person	
China	Administrative Geographical Area	
San Francisco	Administrative Geographical Area	
United States of America	Administrative Geographical Area	
MIT	Organization	
Apple	Business/Company	
Dropbox, Inc.	Business/Company	
Y Combinator	Business/Company	
2007	Date	
2014	Date	
Apr-2014	Date	
Dec-2015	Date	

Figure 75 - Dropbox Entities

The Document Classification function will intercept main categories, such as "IT and information technology"; it represents the advanced function respect to the main topics extraction, and exploits vertical rules developed for a specific domain through a specific taxonomy:

id	label	path	frequency	linked data
20000763	Information technology and computer science	Science and technology/Technology and engineering/Information technology and computer science	34.38%	<p>Dropbox is a file hosting service operated by the American company Dropbox, Inc., headquartered in San Francisco, California, U.S. that offers cloud storage, file synchronization, personal cloud, and client software. Dropbox was founded in 2007 by MIT students Drew Houston and Arash Ferdowsi as a startup company, with initial funding from seed accelerator Y Combinator.</p> <p>Dropbox has experienced criticism and generated controversy for issues including security breaches and privacy concerns.</p> <p>Dropbox has been blocked in China since 2014.</p> <p>Dropbox brings files together in one central place by creating a special folder on the user's computer. The contents of these folders are synchronized to Dropbox's servers and to other computers and devices where the user has installed Dropbox, keeping the same files up-to-date on all devices. Dropbox uses a freemium business model, where users are offered a free account with set storage size, with paid subscriptions available that offer more capacity and additional features. Dropbox Basic users are given two gigabytes of free storage space. Dropbox offers computer apps for Microsoft Windows, Apple macOS, and Linux computers, and mobile apps for iOS, Android, and Windows Phone smartphones and tablets. In March 2013, the company acquired Mailbox, a popular email app, and in April 2014, the company introduced Dropbox Carousel, a photo and video gallery app. Both Mailbox and Carousel were shut down in December 2015, with key features from both apps implemented into the regular Dropbox service. In October 2015, it officially announced Dropbox Paper, its collaborative document editor.</p>

Figure 76 - Dropbox IT Category

Another example of category result, "Shipping services", is shown below:



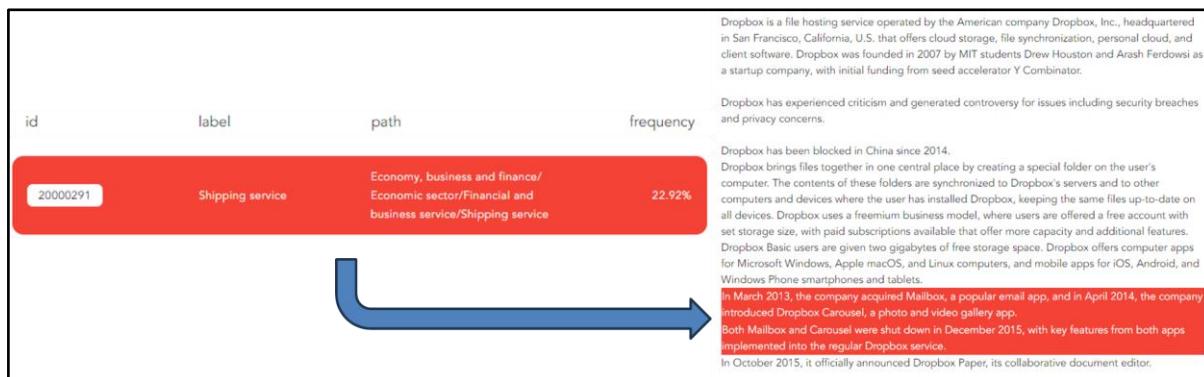


Figure 77 - Dropbox IT Category

The last described analysis has been performed by implementing linguistic rules based on the IPTC taxonomy.

5.6 Monitoring, Reporting and Analytics

This subsystem deals with monitoring, reporting and analytics capabilities on business-level KPIs. It is intended to be used by different DOME user types, like the service providers and consumers, federated marketplaces operators and the DOME operator. It allows the creation of datasets out of different database data sources, which are either located on the DOME infrastructure or on the federated Marketplaces infrastructure. These datasets can be further refined and analysed, and then visualised in a number of different appropriate types of graphs and charts. The visualisations can be then combined into dashboards, according to the user's preferences. The dashboards can be also shared across users, as appropriate.

The scope of the tool's use for the different user types is as follows:

- The DOME administrators can view visualisations related to the negotiations, purchases, conflicts, popularity of services, etc.
- The Service Providers can view visualisations relevant to their own services, on the one side related to negotiations and conversions to purchases, popularity of their services, while they can also -optionally- add their own custom metrics coming from their infrastructure related to their specific services and other providers' related services. While it is expected that service providers will have their own monitoring solutions, the integration of data coming from different sources allows DOME to act as a central monitoring location for scattered services, and also allow them to make these metrics available to their customers.
- Service customers can view the metrics of their purchased services, coming from different service providers, but all accessible through DOME.

The main functionalities of the Monitoring, reporting and analytics tool include:

- Adding data sources and specifying datasets.
- Creating data visualisations, in a wealth of different kinds of visualisations, with customisable data queries.
- Composing data visualisations into different monitoring dashboards.

- Sharing datasets, data visualisations and monitoring dashboards with other DOME users.
- Importing and exporting through web user interface and APIs.
- Preconfigured DOME KPIs.

The application utilises the open source software Apache Superset (<https://superset.apache.org/>). As such many features are already implemented.

5.6.1 Description of interactions with other components

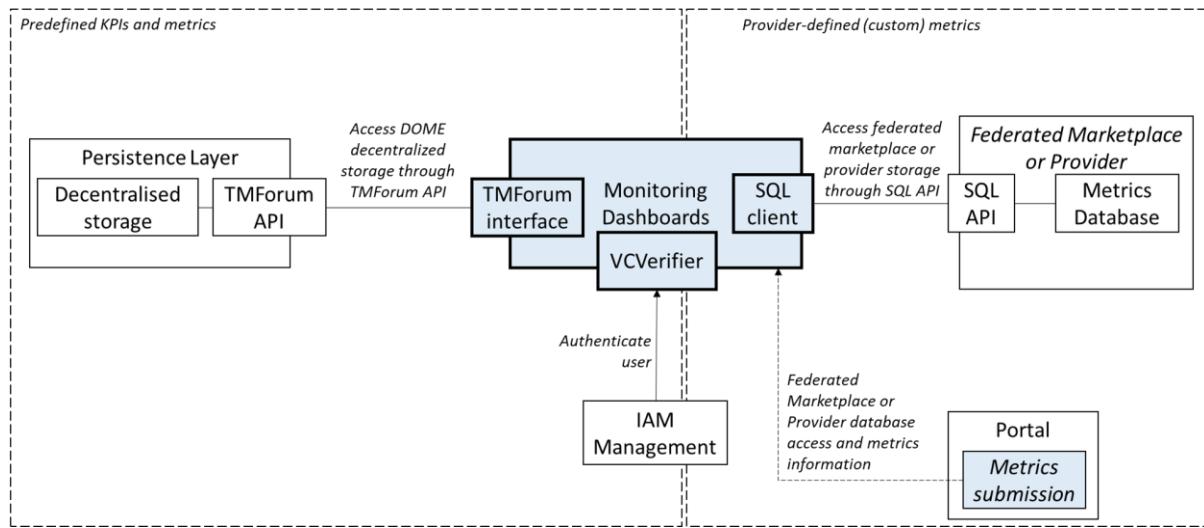


Figure 78 - Main interactions of Monitoring Dashboards with other DOME components

Monitoring dashboards access data from two locations:

1. The DOME decentralised storage, to provide the monitoring of predefined KPIs and metrics that DOME stores according to the TMForum specification.
2. The federated marketplaces or providers, to provide custom-defined metrics with data that is stored on their own databases, which do not participate in the DOME data federation (are not available in the TMForum specification).

The component uses the IAM Management to authenticate the user within the component, through a 'VCVerifier' module.

Additional SQL APIs may be used to access data stores for custom metrics.

The DOME administrator can add, modify and delete the preconfigured KPIs.

For custom metrics, the federated marketplace operator or the service provider uses the portal to submit a template for configuring the access and service metrics for data that does not participate in the data federation.

5.6.2 Creating the monitoring dashboards

5.6.2.1 Data source

'Data source' refers to a data store that can be accessed via an SQL API. There are two types of data sources in DOME:

- Data source that is managed by DOME, and
- Data sources that are managed by the federated marketplaces or service providers, and which do not federate their data in the DOME data federation. Access to these data stores may be useful to customise service-specific metrics.

Only the DOME administrator is able to add a data source. The configuration for accessing the Persistence Layer storage is done only once, and does not need to be modified. The DOME administrator is able to add more data sources if needed.

Figure 79 - Adding a data source (DOME administrator only)

The federated marketplace operator and the service provider are not able to add data sources directly on the monitoring dashboard. Instead, they can submit data store access information and suggested metrics via using the portal. The DOME administrator reviews the submitted information, and approves the addition of the data source to the monitoring dashboards. If the data source conflicts with any of the DOME terms and conditions, the addition of the data source may be rejected.

5.6.2.2 Dataset

Data sets are defined from the data sources and specific schemas that the user has access to. The DOME operator sets up the datasets for the preconfigured KPIs. The federated



marketplace operator and the service provider can also set up datasets within their own data sources.

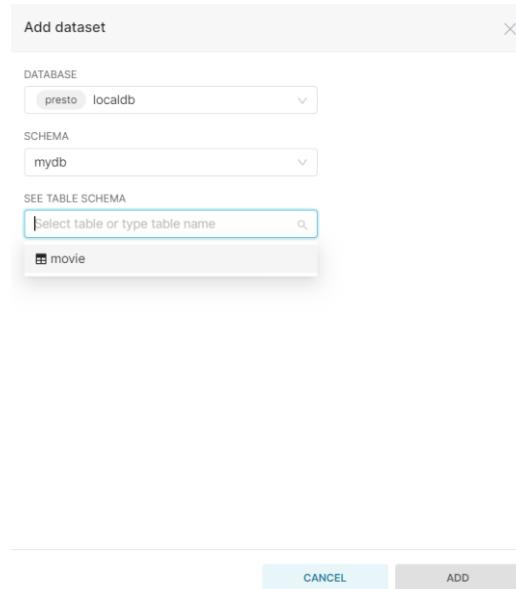


Figure 80 - Adding a dataset out of a data source.

5.6.2.3 Data visualisation

A user that has access to a data set can create their own visualisations in the form of various kinds of charts, depending on the purpose of visualisation. These include: line charts, correlation, gauges, numbers, pie charts, bubble charts, heatmaps, location maps, histograms, area charts, tree charts, etc.

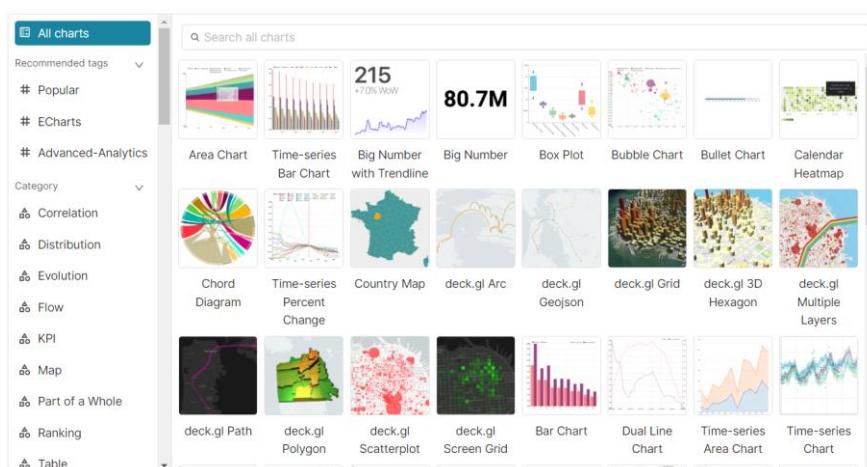


Figure 81 - Examples of available data visualisation chart types.



After selecting a data set and the chart type, the user is able to define the query for the data visualisation using graphical elements and menus, that assist the query structuring by showing the available options. The user is also able to modify the SQL query directly, for more advanced processing. The DOME administrator pre-configures the charts for the preconfigured KPIs, however users are also able to duplicate the existing charts and create their own visualisations on the preconfigured datasets.

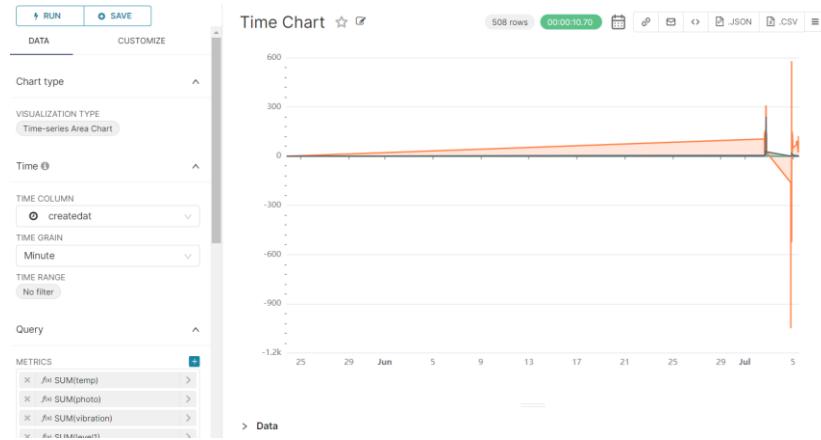


Figure 82 - Configuring the data visualisation chart.

5.6.2.4 Monitoring dashboard

A user can compose the created data visualisation (graphs) into customised monitoring dashboards.

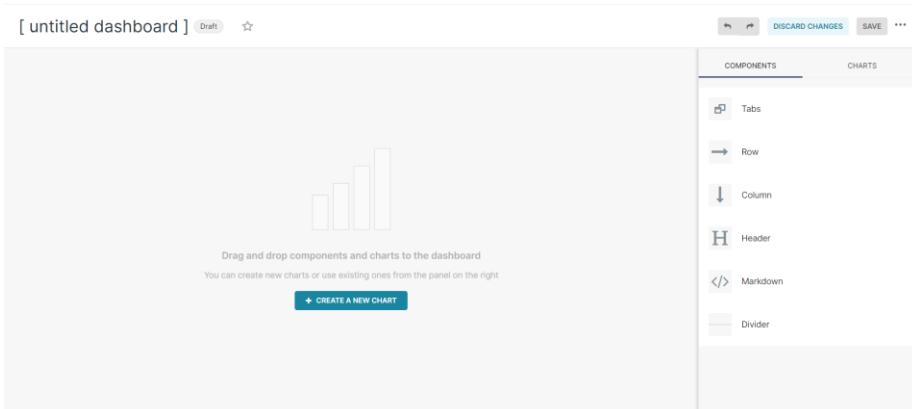


Figure 83 - Creating a new monitoring dashboard.

The users can view a list with the charts they have access to, and by drag & drop they can add the chart to their dashboard.



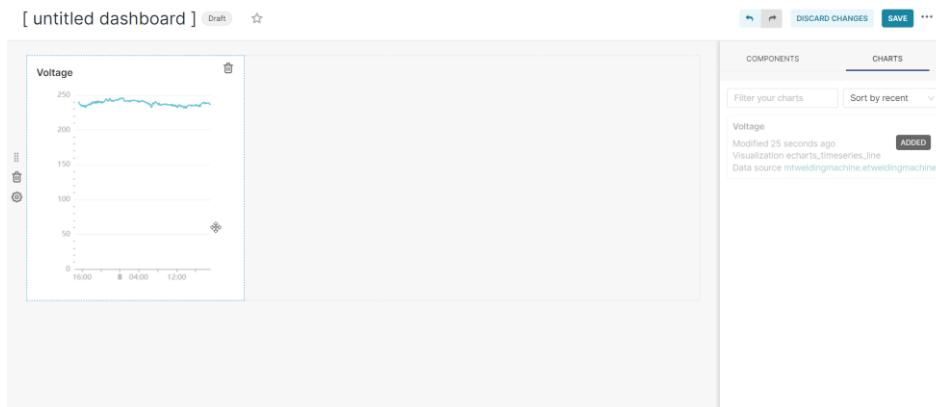


Figure 84 - Adding a chart in the dashboard by drag&drop.

The positioning and the dimensions of each chart can be modified, while it is also possible to add graphical elements like titles and markup.

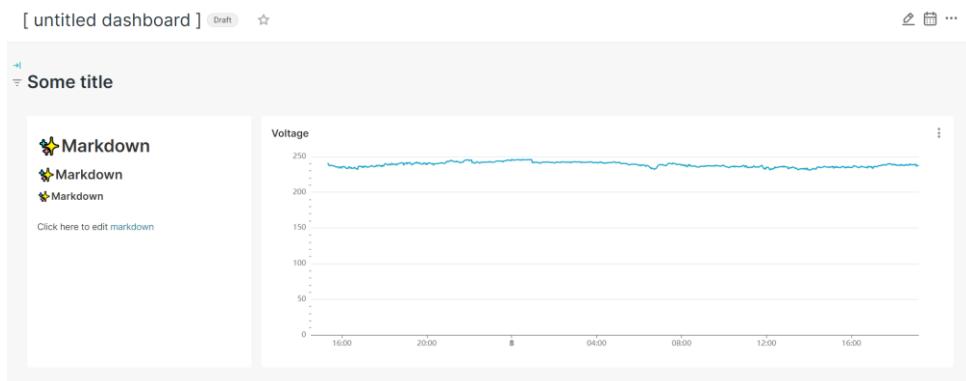


Figure 85 - Modifying charts in a dashboard and adding graphical elements.

5.6.3 Accessing monitoring dashboards

5.6.3.1 View and edit a monitoring dashboard

The user can view the list of dashboards they have access to, i.e., the dashboards they have created and the dashboards that have been shared with them.



Dashboards							
OWNER	CREATED BY	STATUS	FAVORITE	CERTIFIED	SEARCH	BULK SELECT	+ DASHBOARD
[untitled dashboard]	Select or type a value						
[untitled dashboard]		Draft	18 minutes ago				
[untitled dashboard]		Draft	31 minutes ago				
[untitled dashboard]		Draft	27 days ago				
[untitled dashboard]		Draft	28 days ago				
[untitled dashboard]	Untitled Dashboard	Draft	28 days ago				

Figure 86 - List of accessible dashboards.

When opening a dashboard, it is launched in ‘view’ mode. By clicking the relevant icon, the user can directly start editing the dashboard, i.e., change the formatting, modify the charts included, add other elements, etc.

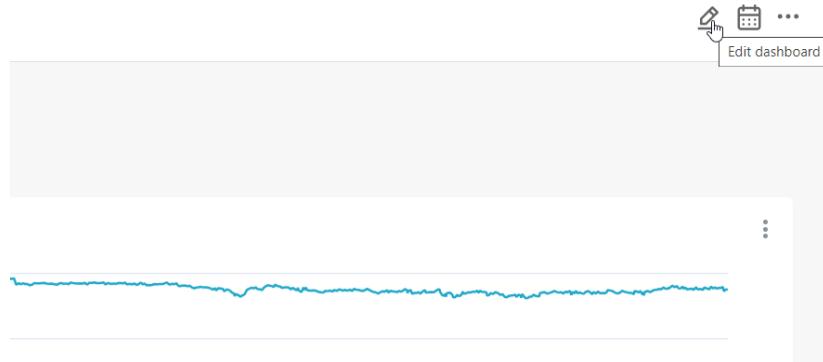


Figure 87 - Start editing an existing dashboard.

5.6.3.2 Managing permissions and sharing

The tool has its own internal specialised permissions management.

The DOME administrator can specify permissions in terms of access to specific functionalities, data querying and data access to users and user groups. Permissions for accessing a limited set of functionalities can also be assigned to all the users, e.g. for demo purposes.

Permissions	<input type="checkbox"/> can read on SavedQuery <input type="checkbox"/> can write on SavedQuery <input type="checkbox"/> can read on CssTemplate <input type="checkbox"/> can write on CssTemplate <input type="checkbox"/> can read on ReportSchedule <input type="checkbox"/> can write on ReportSchedule <input type="checkbox"/> can read on Chart <input type="checkbox"/> can write on Chart <input type="checkbox"/> can read on Annotation <input type="checkbox"/> can write on Annotation <input type="checkbox"/> can read on Dataset <input type="checkbox"/> can write on Dataset <input type="checkbox"/> can read on Dashboard <input type="checkbox"/> can write on Dashboard <input type="checkbox"/> can read on Database <input type="checkbox"/> can read on Query <input type="checkbox"/> can this form get on ResetMyPasswordView <input type="checkbox"/> can this form post on ResetMyPasswordView <input type="checkbox"/> can this form get on UserInfoEditView <input type="checkbox"/> can this form post on UserInfoEditView <input type="checkbox"/> can userinfo on UserOIDModelView <input type="checkbox"/> can list on RegisterUserModelView <input type="checkbox"/> can show on RegisterUserModelView <input type="checkbox"/> can delete on RegisterUserModelView <input type="checkbox"/> can get on OpenApi <input type="checkbox"/> can show on SwaggerView <input type="checkbox"/> can get on MenuApi <input type="checkbox"/> can list on AsyncEventsRestApi <input type="checkbox"/> can invalidate on CacheRestApi <input type="checkbox"/> can export on Chart <input type="checkbox"/> can read on DashboardFilterStateRestApi <input type="checkbox"/> can write on DashboardFilterStateRestApi <input type="checkbox"/> can read on DashboardPermalinkRestApi <input type="checkbox"/> can write on DashboardPermalinkRestApi <input type="checkbox"/> can export on Dashboard <input type="checkbox"/> can get embedded on Dashboard <input type="checkbox"/> can delete embedded on Dashboard <input type="checkbox"/> can export on Dataset <input type="checkbox"/> can read on ExploreFormDataRestApi <input type="checkbox"/> can write on ExploreFormDataRestApi <input type="checkbox"/> can read on ExplorePermalinkRestApi <input type="checkbox"/> can write on ExplorePermalinkRestApi <input type="checkbox"/> can delete on FilterSets <input type="checkbox"/> can list on FilterSets <input type="checkbox"/> can edit on FilterSets <input type="checkbox"/> can add on FilterSets <input type="checkbox"/> can import on ImportExportRestApi <input type="checkbox"/> can export on ImportExportRestApi <input type="checkbox"/> can export on SavedQuery <input type="checkbox"/> can list on DynamicPlugin <input type="checkbox"/> can show on DynamicPlugin <input type="checkbox"/> can query on Api <input type="checkbox"/> can query form data on Api <input type="checkbox"/> can time range on Api <input type="checkbox"/> can this form get on CsvToDatabaseView <input type="checkbox"/> can this form post on CsvToDatabaseView <input type="checkbox"/> can this form get on ExcelToDatabaseView <input type="checkbox"/> can this form post on ExcelToDatabaseView <input type="checkbox"/> can this form get on ColumnarToDatabaseView <input type="checkbox"/> can this form post on ColumnarToDatabaseView <input type="checkbox"/> can external metadata on Datasource <input type="checkbox"/> can external metadata by name on Datasource <input type="checkbox"/> can save on Datasource <input type="checkbox"/> can get on Datasource <input type="checkbox"/> can get value on KV <input type="checkbox"/> can store on KV <input type="checkbox"/> can my queries on SqlLab <input type="checkbox"/> can copy dash on Superset <input type="checkbox"/> can csrf token on Superset <input type="checkbox"/> can fav dashboards by username on Superset <input type="checkbox"/> can profile on Superset <input type="checkbox"/> can validate sql json on Superset <input type="checkbox"/> can csv on Superset <input type="checkbox"/> can fav slices on Superset <input type="checkbox"/> can queries on Superset <input type="checkbox"/> can results on Superset <input type="checkbox"/> can recent activity on Superset <input type="checkbox"/> can available domains on Superset <input type="checkbox"/> can sqlab history on Superset <input type="checkbox"/> can annotation json on Superset <input type="checkbox"/> can tables on Superset <input type="checkbox"/> can request access on Superset <input type="checkbox"/> can add slices on Superset <input type="checkbox"/> can slice json on Superset <input type="checkbox"/> can dashboard on Superset <input type="checkbox"/> can import dashboards on Superset <input type="checkbox"/> can testconn on Superset <input type="checkbox"/> can select star on Superset <input type="checkbox"/> can favstar on Superset <input type="checkbox"/> can search queries on Superset <input type="checkbox"/> can sqlab viz on Superset
-------------	---

Figure 88 Part of permissions that the DOME administrator can assign to users and user groups.

On the other hand, each user can specify access to their own datasets, charts and dashboards. In such a scenario, data visualisations can be shared across different user groups within DOME, like a service provider and a service consumer. It needs to be noted that the authorization is verified in all the steps from the dashboard to the dataset. For example, to view a dashboard that contains a chart, the user needs to have access to the dashboard, the chart and the dataset.



AUTOCOMPLETE QUERY PREDICATE

1 |

EDIT SQL IN MODAL

When using "Autocomplete filters", this can be used to improve performance of the query fetching the values. Use this option to apply a predicate (WHERE clause) to the query selecting the distinct values from the table. Typically the intent would be to limit the scan by applying a relative time filter on a partitioned or indexed time-related field.

EXTRA

1 |

Extra data to specify table metadata. Currently supports metadata of the format: '{ "certification": { "certified_by": "Data Platform Team", "details": "This table is the source of truth.", "warning_markdown": "This is a warning." }'.

OWNERS

[REDACTED]

TEMPLATE PARAMETERS

A set of parameters that become available in the query using Jinja templating syntax

USE LEGACY DATASOURCE EDITOR CANCEL SAVE

(1)

Basic information

* Name
Voltage

Description

The description can be displayed as widget headers in the dashboard view. Supports markdown.

Certification

Certified by

Person or group that has certified this chart.

Certification details

Any additional detail to show in the certification tooltip.

Configuration

Cache timeout

Duration (in seconds) of the caching timeout for this chart. Note this defaults to the dataset's timeout if undefined.

Access

Owners

A list of users who can alter the chart. Searchable by name or username.

CANCEL SAVE

(2)



The screenshot shows the 'Dashboard properties' dialog box with the 'Basic information' tab selected. It contains the following fields:

- TITLE:** [untitled dashboard]
- URL SLUG:** (empty)
- Access:** OWNERS: A dropdown menu showing a single entry.
- Colors:** COLOR SCHEME: A dropdown menu labeled 'Select scheme'.
- Certification:** CERTIFIED BY: A dropdown menu showing a single entry. CERTIFICATION DETAILS: A text input field containing placeholder text.
- ADVANCED**: A link to expand advanced settings.
- CANCEL** and **SAVE** buttons at the bottom.

(3)

Figure 89 - A user can be granted access to a dataset (1), chart (2) and dashboard (3) by adding them in the 'OWNERS' fields

5.6.4 Importing, exporting and APIs

5.6.4.1 Import and export via web UI

The web interface provides access to importing data in the form of .csv, columnar or excel files. The user must already have access and write permissions to an existing database, where the imported data will be written. The user must specify the database to be used, the table to be created and the database schema.



Columnar to Database configuration

Table Name *	Table Name Name of table to be created from columnar data.
Columnar File *	<input type="button" value="Choose Files"/> No file chosen Select a Columnar file to be uploaded to a database.
Database	examples ▾
Schema	test ▾ Specify a schema (if database flavor supports this).
Table Exists *	<input type="button" value="Fail"/> Fail If table exists do one of the following: Fail (do nothing), Replace (drop and recreate table) or Append (insert data).
Use Columns	null Json list of the column names that should be read. If not None, only these columns will be read from the file.
Dataframe Index	<input type="checkbox"/> Write dataframe index as a column.
Column Label(s)	Column Label(s) Column label for index column(s). If None is given and Dataframe Index is True, Index Names are used.

Figure 90 - Importing data through .csv, excel or columnar file

The user can also export the data of a chart through the chart editing section. The chart values can be exported into .json or .csv format. The export to image of the chart is also available.

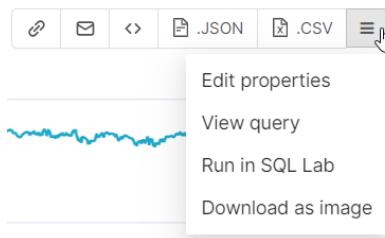


Figure 91 - Export options for a chart

5.6.4.2 Monitoring dashboard-specific API

The tool has also an API for executing advanced functions. Access to this API will be provided only to authorised users or federated marketplace operators. The API follows the openAPI specification and provides a web page ('swagger') where all the available functions can be viewed, along with instructions on how to execute the API calls and the expected responses. Among others, the API calls allow the authorised user to import and export database, dataset, charts and dashboard configurations, as well as get list, create and delete charts, datasets, dashboards, filters and queries.



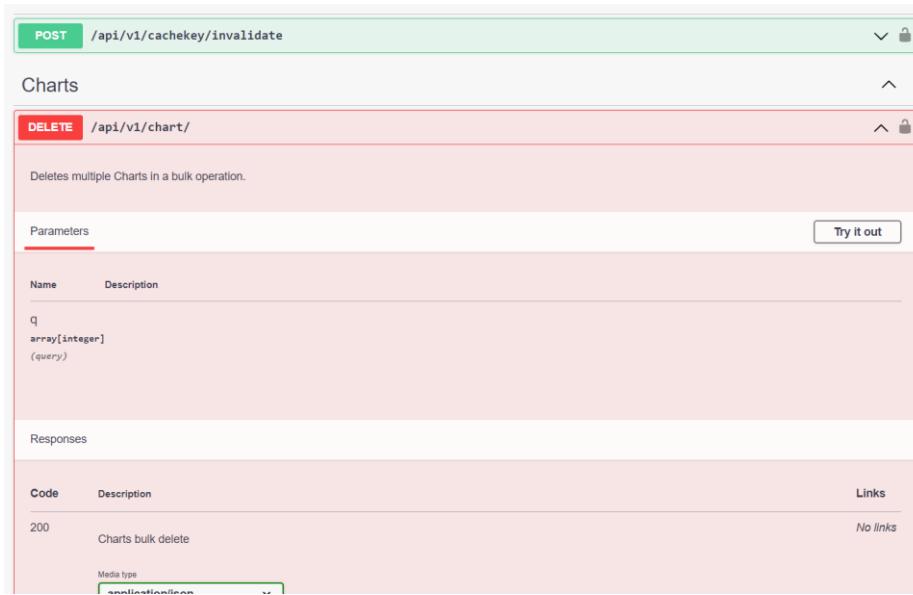


Figure 92 - Web page with all the available API calls, instructions on calls and responses

5.6.5 Preconfigured KPIs

The DOME admin can pre-configure metrics (KPIs) that are by default available to the different DOME user types. While these metrics can be configured via the Graphical User Interface, a set of indicative KPIs are given here for complementarity. These may be easily modified and extended during the DOME operation, according to new needs identified.

5.6.5.1 All

This paragraph described the preconfigured KPIs that are available to any DOME user.

- Number of Service providers
 - Total
 - Registered directly in DOME - from federated Marketplace (and which)
- Average number of Service providers per year quarter
 - Registering in DOME
 - Added from federated Marketplace (and which)
 - Requesting verification
 - Getting verified
- Average time for service provider to get verified
 - From registration to requesting verification
 - From requesting verification to being verified
 - From registration to being verified
- Average number of Service providers per year quarter
 - Added from federated Marketplace (and which)
 - Getting verified
- Number of offered services

- Total
- Catalogued directly in DOME - from federated Marketplace (and which)
- Average per Service Provider
- Per type of service

5.6.5.2 Service consumer

This paragraph describes the additional preconfigured KPIs for the service consumer.

- Complaints
 - Average time for ticket to move from open to resolved

5.6.5.3 Service provider

This paragraph describes the additional preconfigured KPIs for the service provider.

- Average number of Service providers per year quarter
 - Requesting verification
- Average time for service provider to get verified
 - From registration to requesting verification
 - From requesting verification to being verified
 - From registration to being verified
- Complaints
 - Average time for conflict resolution ticket to move from open to resolved
- Number of visits to own profile and per offered service

5.6.5.4 Federated marketplace provider

This paragraph describes the additional preconfigured KPIs for the federated marketplace provider.

- Average number of Service providers per year quarter
 - Registering in DOME
- Price of services
 - Total average
 - Average per service type
- Service sales
 - Number of services sold through DOME
 - Total value of specific's marketplace services sold through DOME
 - Average value of specific's marketplace services sold through DOME
 - Average value of specific's marketplace services sold through DOME, per service type
- Complaints
 - Number of conflict resolution tickets for Service providers coming from the specific federated marketplaces and status
- Web portal statistics
 - Number of visitors on portal
 - Aggregation of visits to service and service providers per federated marketplace

5.6.5.5 DOME operator

DOME operator views all of the above metrics of all user types, and in addition:



- Number of Service providers, and
 - Verified - non verified
 - Removed
- Average number of Service providers per year quarter
 - Registering in DOME
- Service sales
 - Number of services sold through DOME
 - Total value of services sold through DOME, per service provider directly registered in DOME and federated marketplace, and which.
 - Average value services sold through DOME, per service provider directly registered in DOME and federated marketplace, and which.
 - Average value services sold through DOME, per service type
- Number of complaints
 - Total number of conflict resolution tickets and per status
 - Number of conflict resolution tickets per service provider and status
 - Number of conflict resolution tickets for service providers coming from federated Marketplace (i.e., per federated Marketplace) and status
- Web portal statistics
 - Source of visits
 - Number of new registrations and per user type
 - Number of page visits and aggregated popularity list
 - Number of visits to specific service provider profiles and aggregated popularity list
 - Number of visits to specific services and aggregated popularity list
 - Average time spent

5.7 User Support

5.7.1 General Concepts

5.7.1.1 Focus of the customer service

The CUSTOMER SERVICE focuses on solving any kind of need the portal user may express in relation to the DOME ecosystem).

While this excludes the direct support on services provided by 3rd parties (Providers) through the DOME catalogue, in charge of the specific provider, it considers the management of requests about such a product but only to forward those questions to the right response centre.

This excludes all backoffice maintenance activities that will be addressed as OPERATIONS PROCEDURES in another task of the project (ex.: Federating a new marketplace, creating new categories on the portal, integrating a new payment gateway and so on).

In terms of contents the customer service will focus on the DOME portal usage/usability.

This includes following features:

- User guide (how to perform specific actions/operations)
 - Troubleshooting guide (how to address malfunctions)
-



- Recovery guide (how to recover from wrong operations)

The above features will be designed to cover the needs of following kind of users:

- Buyers (any categories)
- Publishers (catalogue owners)

A limited support (documental knowledge base only) will be provided to those content categories:

- Integration services (API documentation)

No support is actually expected to be provided on following contents:

- Usage and troubleshooting of Product/services procured on the catalogue (support in charge to the product owner)

5.7.1.2 Roles/people involved in customer support

As we previously said the scope of the customer support is to help the portal users (mainly buyers and sellers) in addressing any kind of issue (information query or malfunction management) related to the DOME portal.

The operational model we're aiming to implement sees a first level team acting as primary contact point and a second level support acting as specialistic competence centre.

We're aiming to solve at first level any kind of info request through self-service utilities (an online help system) and through an automated chatbot, able to interact in natural language and specifically trained to answer about the DOME portal. We strongly rely on the capability of the chatbot automation to satisfy the customer requests and so the usage of first level human operators will be very limited. Both the 1st level support operator and the chatbot will be trained on a common knowledge base containing all the needed information about the platform usage and functionalities.

A different approach needs to be put in place to build the 2nd level support team.

This team, during the project development time and until the DOME operator will not take care of the service, will be built aggregating all the competence centres that will contribute in the creation of the different portal modules and functionalities in order to be able to manage any kind of troubleshooting request coming from the real portal users. The 2nd level support team is committed to building and improving the platform knowledge base the 1st level will rely on.

5.7.1.3 Language

All the DOME portal contents will be provided in English, including the customer support interface.

The introduction of the capability to handle multi language contents will be lately evaluated as an optional enhancement of the platform.

5.7.1.4 Managed Data

The customer service scope is the portal functionalities usage. This means that the 90% of the handled data is generic technical data describing the portal functionalities and the operation process. In some case the Customer service platform may touch some customer referred data:



1. Business data related to the submitted orders, their billing, the kind of subscribed services and so on.
2. Personal data mainly related to contact needs (phone number, shipment address, email contact etc.) or business/payment needs (bank/credit card accounts, legal entity name, company role, and so on).

Being not sensitive data information, there are no specific treatment requirements but as an additional customer care policy all such data will be encrypted both at rest or in transit during all the customer interactions.

The chatbot will be instructed to avoid the exploitation of any personal data just reporting the customer the link of the management function where he can find the requested information in a private protected panel.

5.7.2 Customer support technical platform

The customer can engage the customer support through on-line services only. Phone or email communication may be activated, upon needs, by the customer support team to provide the requested assistance but not as an entry point. The service will rely on 3 main tools that are described hereafter.

5.7.2.1 Knowledge base

A structured knowledge base will be provided as a complementary component of the portal baseline to collect all the relevant information to support the day-by-day operations of the various portal user categories. The knowledge base contents will be provided by all the project members contributing in the building of the portal functionalities.

The platform to handle the knowledge base has not been yet identified but following requirements have been already pointed:

- The platform must be freely accessible by the customers through the chatbot.
- Contents will be filtered upon the user type: Guest/Authenticated.
- The platform must be accessible by the customer support team (operators) through a web interface or the chatbot.
- The knowledge base contents reference language will be English. The capability to handle multilingual contents will be evaluated lately as an optional feature.

One of the primary criteria we are trying to address is the simplification of the knowledge base content publishing. Being an effort consuming task as much as we can simplify it as much content we will have from the contributors.

The simplest solution is the creation of a blob repository full of readable documents that can be indexed by the chatbot that will lately re-generate the text in a readable format for the user.

The Knowledge base solution should rely on a high availability platform able to ensure business continuity and data integrity.

5.7.2.2 Chatbot

An AI based chatbot will allow the customer to perform any kind of query/investigation on the knowledge base information base and to ask for support in case of issues or malfunctions.

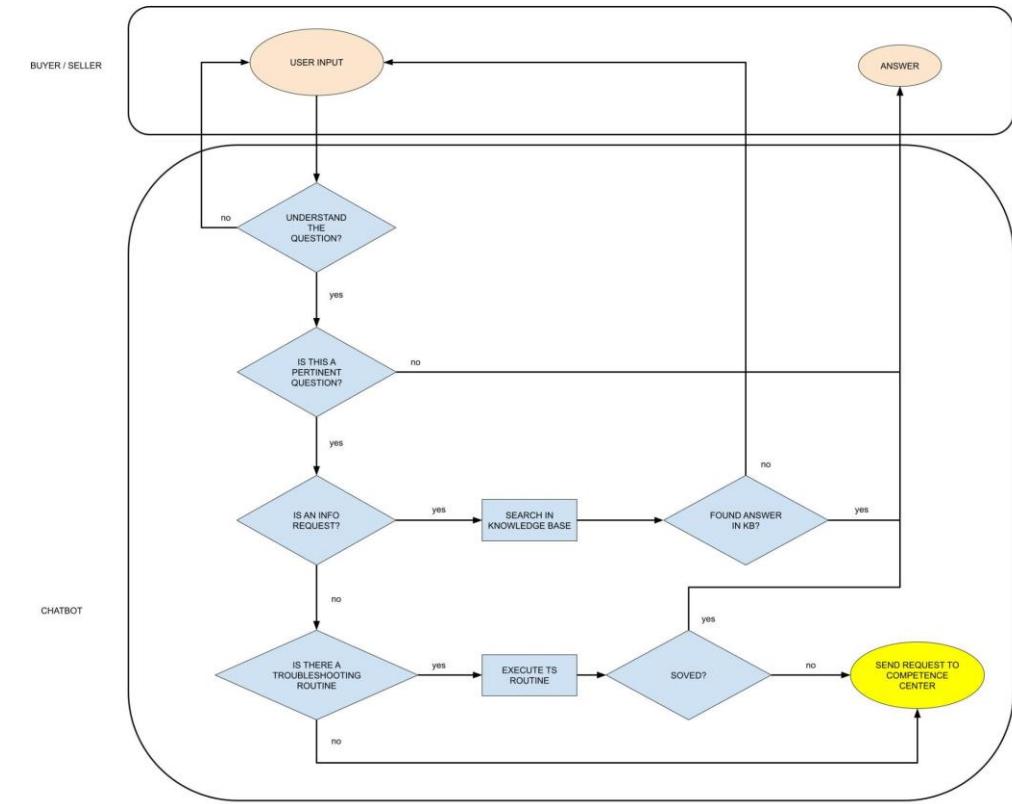


The chatbot will act as a standard help desk operator, dealing through natural language, understanding the customer request and providing readable answers. In some cases it can interact with the portal API to gain specific information through the platform. This will be based on a proprietary Engineering DHUB tool (GPT 3.5 enabled), that is part of our managed services tools suite, vertically customised and trained to answer about the portal usage and troubleshooting.

The baseline to train the chatbot will be the Knowledge base platform. This means that adding new contents to the knowledge base will enable the capability to extend the knowledge of the chatbot.

The chatbot will provide more specific assistance to registered users and more general information to guest users visiting the website (user category weight) and will be equipped with “filtering” capabilities in order to avoid answering some specific categories of questions.

Note: being not part of the core components of the portal this is outside the scope of the open source released packages.

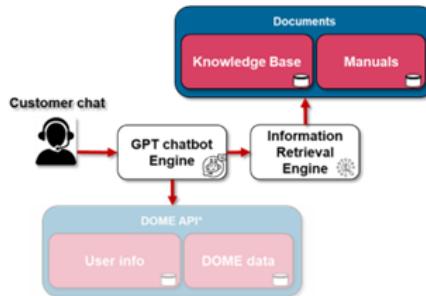


This solution is designed to work without accessing any data or general information outside the DOME marketplace perimeter. This means that all data exchanged between users and the chatbot still remain in our infrastructure and there aren't any other 3rd party tools involved in the entire data elaboration process.

The platform architecture will include following components:

- GPT chatbot engine: the conversational engine that communicates with the user and is able to understand requests and to query, as needed, the APIs of the store or the

- Information Retrieval engine, which will be created through fine tuning on specific vertical documentation on the subject.
- Information Retrieval Engine: it is the engine that extracts from the documents the information useful for responding to user requests



- Documents: it is the database containing the information describing the vertical domain in which the chatbot must be an expert.

Note: DOME API are not part of the chatbot implementation

5.7.2.2.1 Input data

The implementation of the chatbot will rely on following inputs:

- Knowledge Base & Manuals: Documentation containing domain information that the chatbot must be familiar with
- Examples of Dome API interactions: the examples of output that the API interface can provide to the chatbot when the conversational engine decides to query the platform.

5.7.2.2.2 Provided functionalities

- Ability to understand natural language user requests
- Ability, based on user requests, to execute flows of actions (or workflows) described by the use cases identified in the analysis phase
- Ability to extract information from the document base to answer user questions
- Ability to query Dome APIs to perform useful actions to respond to user requests

5.7.2.3 Troubleticketing system

Where the chatbot will not be able to provide the correct feedback to the user, it will rely on a physical person (technical operator) that will perform the investigation on his behalf. This kind of activity is usually tracked through a troubleticketing system able to keep tracking of the request, of the provided answer and of the global performance of the support service (time to solve the issues).

The involvement of human operators may occur at 1st level (the user may ask the chatbot to talk directly with an operator) or in case of malfunctions requiring the intervention of a 2nd level competence centre.

To track the support activities an ITIL compliant troubleticketing system will be introduced as a baseline platform. This platform must ensure traceability and measurability of the activities performed by the customer support team.

Assuming the involvement of a future DOME operator as owner of the Customer support service this can be his own original troubleticketing system.

Until the appointment of the DOME operator a dedicated troubleticketing system must be provided.

The troubleticketing system must rely on a High Available platform ensuring business continuity, data preservation, and security.

While the knowledge base will handle potentially all the relevant information query raised by the customer, the trouble ticketing system has the mandatory responsibility to enable the customer requesting support for custom cases or for situations that he has not been able to address and solve through the knowledge base. In this sense the availability of a communication channel allowing the capability to simply describe in natural language the support request is a mandatory feature that we need to provide since the first release of the platform. It can be a simple web form or an email address, and later, according to the evolution of the platform, can grow up to a multi-channel communication service if needed.

Enabling a communication channel with the user is not only a matter of letting him express the request, but also to trace all the customer contact. This tracing capability enables the capability to later review those requests better understanding potential service weakness and to define a continuous improvement strategy. While a first draft version of the service, to be used in the early stages of the development, can be based on a simple input form, to have an effective and efficient collection of the user request it's important to have the capability to classify at first contact the user request. As baseline the request must be tagged with following attributes:

- Context. The identification of the area the request is referring to (ex. Billing or catalogue browsing) allows the capability to focus a search in the knowledge base or to identify the correct competence centre deputed to support a malfunction resolution.
- Typology. The understanding of the nature of the request, so understanding if it's an information request or a request of support for a malfunction helps in defining the correct path to address it.
- Severity. While committed to serve all the requests in the shortest time possible the Customer Service may be in the condition to decide who serves first and this is done through the definition of a severity level (blocking issues are addressed before simple information queries).

As anticipated the above classification is done at first contact and is expressed by the user. This may lead to wrong classifications. The trouble ticketing system may provide the capability for the support operator to re-classify the request upon his professional understanding in order to correctly address it.

Another relevant capability of the trouble ticketing system, while not strictly needed to provide support to the user, is the resolution classification. Having the visibility of the resolution methodology adopted for the resolution of the different cases helps the continuous improvement process. As baseline such classification must be described through the following attributes:

- Issue category. This clarifies the reason for the documented issue. It can be "wrong operation", "product malfunction", "bad data", "missing information". This list may be improved according with the service evolution.



- Fixing methodology. This clarifies how the issue has been fixed. It can be “integrated guideline”, “code fix”, “data fix”. This list can be lately being improved according with the evolution of the service.

Note: being an internal usage functionality the resolution classification will be shown in the operator dashboard but not on the User dashboard.

Workflow management.

Any technical activity should be performed following a reference procedure. The user case resolution is not an exception to that paradigm. In this sense the possibility to pre-define management workflows that forces the first contact classification, automatically address the request to the right competence centre, and so on is a way to improve the global quality of the service and to ensure the respect of the predefined management procedures.

The workflow design should be addressed through a configuration tool, if possible codeless allowing the customer support team to adapt the different flows upon the service evolution without modifying the trouble ticketing engine.

Self service portal

While the first request can reach the Customer Service through a multi channel service, the best way to provide visibility of the management of the request inside the customer support organisation is a web self service portal not only including the input form to enter the request, but supported by some dashboard showing the status of the request, the classification, the expected resolution time and, once done the resolution comment. This helps the customer to autonomously and continuously monitor the status of his request. The same dashboard may optionally include the capability to request for a direct update where a customer service deputy will contact the user to update him about the request status..

As a baseline we will address the web interface only. It's possible, if requested by the users, to introduce a mobile version of the user interface in a second phase of the project.

Operator dashboard

The operator dashboard has a web interface that allows the support operator to have visibility of the assigned requests, their classification, and the SLA applied to the single case. Through that dashboard the operator can update the use case up to his final resolution (including resolution classification) or can forward the request to another competence centre.

The same dashboard will provide some additional statistical information about the number of tickets in the queue or about the remaining time to reach the SLA threshold.

The operator dashboard will not have a mobile interface.

5.7.3 Service level and KPIs

5.7.3.1 Service Level Agreement

As baseline design the Customer service front-end (contact point) will be represented by the integrated chatbot that must ensure proper availability to enable the user to enter support requests + some human operator to integrate the chatbot offload.



This will land in a first KPI of 98% of availability on a monthly basis in a 7x24 availability window.

In some cases the 1st level support will not be able to provide the desired support and the call must be forwarded to a 2nd level support technician that can deal with the request.

For such cases a second KPI (first contact time) has been defined: time to take care of the call = 2hr in a 8x5 availability window.

Due to the different request typologies (from simple information query to troubleshooting processes) it's not possible to define a fixed resolution time valid for the different categories. Due to this, we're not going to define this kind of KPI.

5.7.4 Sample Use case

The chatbot will be, in the first instance, implemented to answer questions referring to marketplace documentation. More specifically: the chatbot will be used by users to get clarification on the use of the portal but also for the services that are able to deliver.

5.7.4.1 Query categories

This paragraph describes the different query categories the chatbot will address. This defines the list of algorithms the chatbot must implement to ensure proper functionality and the content typology the knowledge base must provide to instruct the chatbot. A final category named "Forbidden queries" will describe the kind of answers the chatbot will never answer to protect personal data and comply with ethical rules.

- How to – The typical case is the user guide where the user asks “how to” perform some action. The default source of information is the knowledge base that should include contents to answer such kinds of questions. The answer generally describes a user operation flow to achieve the desired result (can be a simple link to a specific procedure already described in the knowledge base).

Some sample:

- How can I change my password?
 - How can I add some resources to a provisioned service?
 - How can I change the plan of the service MongoDB? I want to pay the service flat and not yet the pay per use
 - I bought a service. I don't find instructions on how to use it. How can I perform the access?
 - How it works – These questions are related to knowing how the DOME portal, or some specific workflows, works.
- Some sample:
- Which payment methods are allowed? Is it possible to pay without the credit card?
 - How must I wait before using a service after the payment?



- Is it possible to update by billing information?
- How to write with a human? Is it possible?
- I've an issue – Every time a user is in trouble and writes to the chatbot, in a way to get a solution or a generic way to solve/manage the issue.

Some sample:

- The instruction that tells how to access service XXX doesn't work, how can I perform the access?
- I wrote 2 times my new password in the section dedicated, but it isn't updated
- I don't remember the password and I can't login, the reset password does not work and I can't sign into DOME. What can I do?
- I lost my ssh private key for access to my VM with ip 1.2.3.4, how can I resume it?
- Forbidden queries – All the questions that do not strictly concern the area of competence for the user and/or sentence that include such kind of personal data not required.

Some sample:

- On which date has the service XXX been published to DOME?
- How many users are registered to DOME?



5.7.4.2 Query samples

This paragraph provides some examples of use cases of interaction between DOME stakeholders (end user & publisher) and the DOME customer assistance (using chatbot like 1st layer of interaction and opening a ticket to the proper competence centre like 2nd layer).

The two tables included below must be completed by all Competence Centers that are involved in the Assistance. Here follow some examples of possible questions and how they are processed in the assistance workflow (Chatbot replies directly or messages are delivered to a specific competence centre).

Customer events	Onstage contact action : 1st level Service Desk - Chatbot	Backstage contact action - Billing competence center	Backstage contact action - Infrastructure competence center	Backstage contact action - Dome competence center	Backstage contact action - Privacy & compliance competence center	Backstage contact action - Account management competence center	Backstage contact action - Products management competence center
		Support process action	Support process action	Support process action	Support process action	Support process action	Support process action
General info request about Dome's portal	Info request identified -> Search in the knowledge base -> creation of the answer						
Billing % invoicing info or issue with troubleshooting routine implemented	Info request identified -> Search in the knowledge	Competence center is not involved if it documents					



	base -> creation of the answer	routine procedures for training the chatbot					
Billing & invoicing issue without troubleshooting routine implemented	Open ticket to specific competence center	Competence center take in charge the ticket, fix the problem and close the ticket					
Infrastructure info or issue with troubleshooting routine implemented	Info request identified -> Search in the knowledge base -> creation of the answer		Competence center is not involved if it documents routine procedures for training the chatbot				
Infrastructure issue without troubleshooting routine implemented	Open ticket to specific competence center		Competence center take in charge the ticket, fix the problem and close the ticket				
Web portal info or issue with troubleshooting routine implemented	Info request identified -> Search in the knowledge base -> creation of the answer		Competence center is not involved if it documents routine procedures for training the chatbot				
Web portal issue without troubleshooting routine implemented	Open ticket to specific competence center		Competence center take in charge the ticket, fix the				



				problem and close the ticket			
Catalog's search info or issue with troubleshooting routine implemented	Info request identified -> Search in the knowledge base -> creation of the answer			Competence center is not involved if it documents routine procedures for training the chatbot			
Catalog's search issue without troubleshooting routine implemented	Open ticket to specific competence center			Competence center take in charge the ticket, fix the problem and close the ticket			
Account management info or issue with troubleshooting routine implemented	Info request identified -> Search in the knowledge base -> creation of the answer				Competence center is not involved if it documents routine procedures for training the chatbot		
Account management issue without troubleshooting routine implemented	Open ticket to specific competence center				Competence center take in charge the ticket, fix the problem and close the ticket		
3rd party App management (provisioning/deprovisioning) info or issue with troubleshooting routine implemented	Info request identified -> Search in the knowledge base ->					Competence center is not involved if it documents routine procedures for	



	creation of the answer						training the chatbot
3rd party App management (provisioning/deprovisioning) issue without troubleshooting routine implemented	Open ticket to specific competence center						Competence center take in charge the ticket, fix the problem and close the ticket
Product management (buyer & seller) info or issue with troubleshooting routine implemented	Info request identified -> Search in the knowledge base -> creation of the answer						Competence center is not involved if it documents routine procedures for training the chatbot
Product management (buyer & seller) issue without troubleshooting routine implemented	Open ticket to specific competence center						Competence center take in charge the ticket, fix the problem and close the ticket
Privacy & compliance info or issue with troubleshooting routine implemented	Info request identified -> Search in the knowledge base -> creation of the answer				Competence center is not involved if it documents routine procedures for training the chatbot		
Privacy & compliance issue without troubleshooting routine implemented	Open ticket to specific competence center				Competence center take in charge the ticket, fix the problem and close the ticket		



API integration info or issue with troubleshooting routine implemented	Info request identified -> Search in the knowledge base -> creation of the answer			Competence center is not involved if it documents routine procedures for training the chatbot			
API integration issue without troubleshooting routine implemented	Open ticket to specific competence center			Competence center take in charge the ticket, fix the problem and close the ticket			

Category	Description	Information queries	Troubleshooting guide
Billing & Invoicing	This category covers all billing and Invoicing-related interactions, such as viewing invoices, troubleshooting billing problems, etc.	General description of the Billing service Modes of calculation provided by the Billing engine Reference currency of the Billing engine Detailed cost analysis of individual services Viewing of invoices Download copy of invoices Payment methods that are accepted for billing Adding new payment method	Incorrect invoice Debit not issued Double chargeback Errors in individual service charges in billing data Inability to view invoices Difficulty downloading a copy of invoices Problems encountered when adding a payment method Inability to set a default payment method



		Setting default payment method	Difficulty viewing transaction history
		Viewing transaction history	Problems encountered with a specific transaction
		Refund requests	Inability to request a refund
		How to contact support for billing issues	Errors found in billing information
		How to update billing information	Difficulty updating billing information
		How to make or unsubscribe from recurring billing	Problems encountered with recurring billing
Product management (buyer & seller)	This category covers all interactions related to the publication of a product or service on the portal.	Publication of a new product on the portal	Problems when publishing a new product
		Requirements for publishing a product	Errors when setting the price
		Setting the price for a product	Difficulties in adding images or descriptions
		Adding images or descriptions to the product	Problems in tracking product sales
		Tracking sales of the product	Difficulties in promoting the product
		Promoting the product on the portal	Problems updating product information
		Updating product information	Difficulties in removing a product from the portal
		Removing a product from the portal	Problems with product visibility on the portal
		Account creation	Problems accessing the account
Account management (buyer & seller)	This category covers all interactions related to account management, such as registration, logging in, editing account details, deleting, etc.	Password recovery	Failure to receive verification email
		Updating account information	Difficulty changing password
		Adding a payment method to the account	Problems adding a payment method
		Changing the password	Errors when updating account information
		Setting up notifications for the account	Difficulties in setting up notifications
		Closing the account	Problems in closing the account
		Verifying the email address or phone number	Difficulties in verifying the email address or phone number



Privacy & Compliance	This category covers all privacy and compliance-related interactions, such as privacy policy inquiries, reports of compliance issues, etc.	Portal privacy policy	Difficulty finding the privacy policy
		Procedure for removing personal information from the portal	Problems in requesting removal of personal information
		Reporting a privacy breach	Difficulties in reporting a privacy violation
		Security measures taken by the portal to protect user data	Problems in accessing personal data on the portal
		Access to personal data stored on the portal	Difficulties in requesting correction of personal data
		Privacy laws and compliance followed by the portal	Problems in filing a privacy-related complaint
		Request for correction of personal data	Difficulties in restricting the use of personal data
		Filing a privacy-related complaint	Lack of notification of privacy policy changes
		Limitation of the use of personal data	Problems with account privacy settings
		Notification of changes to the privacy policy	Concerns about data security on the portal
Catalog	This category covers all interactions related to searching for products or services in the portal catalog, the correct labels of a service in the catalog categories, etc.	Searching for a specific product or service in the catalog	Problems searching the catalog
		Filtering of search results in the catalog	Difficulty filtering search results
		Sorting of search results in the catalog	Problems sorting search results
		Viewing the details of a product in the catalog	Difficulty viewing the details of a product
		Comparing different products in the catalog	Problems in comparing different products
		Viewing reviews of a product in the catalog	Difficulty viewing reviews of a product
		Viewing products similar to the one searched for	Problems viewing similar products
		Requesting additional information about a product in the catalog	Difficulties in requesting additional information about a product
		Services available on the portal	Portal access problems
Web portal		Requirements for becoming a vendor	Problems loading the portal



	This category covers general inquiries about the portal, such as its features, services offered, etc.	Key portal features	Difficulty editing preferences
		Contact with customer support	Problems with email notifications from the portal
		Portal policies regarding returns and refunds	Problems with API integration
		Availability of API integrations	Difficulty in displaying portal content correctly (also UI/UX issues)
		Availability of a mobile app	Problems with the portal mobile app
Infrastructure	This category covers all interactions related to portal infrastructure, such as infrastructure inquiries, reports of infrastructure problems, etc.	Portal infrastructure specifications	Problems accessing the portal infrastructure
		Maintenance of the portal infrastructure	Errors during the use of the infrastructure
		Infrastructure security measures	Difficulties during the maintenance of the infrastructure
		Reporting a problem with the infrastructure	Security problems with the infrastructure
		Data backup and recovery policies	Difficulties with data backup and recovery
		Information on infrastructure scalability	Problems with the scalability of the infrastructure
API integration	This category covers all interactions related to API integration, such as configuration, integration troubleshooting, etc.	Integration of the web portal with a system	Problems accessing the portal API
		Access to the portal API	Errors with the portal API
		Limitations of the portal API	Difficulties in obtaining an API key
		Obtaining an API key	Security problems with the API
		Features available through the API	Difficulties in monitoring the use of the API
		Testing of the portal API	Problems with testing the portal API
		Security requirements for using the API	Difficulties with the limitations of the portal API
		Monitoring the use of the API	Problems in reporting a problem with the API
		Reporting a problem with the API	Problems with web portal integration



		Best practices for using the portal API.	Need for assistance with best practices for using the API
Third-party application management (provisioning/deprovisioning)	This category covers all interactions related to the management of third-party applications (for example other cloud ecommerce that perform federation of their services on the Dome portal), such as installation, uninstallation, configuration, deleting, etc.	Adding a third-party application to the account	Problems in adding a third-party application
		Removing a third-party application from the account	Difficulties in removing a third-party application
		Third-party applications supported by the portal	Problems in configuring a third-party application
		Configuring a third-party application	Difficulties in managing permissions for a third-party application
		Managing permissions for a third-party application	Problems connecting a third-party application to the account
		Connecting a third-party application to the account	Problems with compatibility with a third-party application
		Troubleshooting compatibility issues with a third-party application	Difficulties in obtaining support for a third-party application
		Obtaining support for a third-party application	Problems in updating a third-party application
		Updating a third-party application	Difficulties in monitoring the usage of a third-party application
		Monitoring the usage of a third-party application	Problems with integrating a third-party application



6 Appendix I: example of onboarding of organisations in DOME

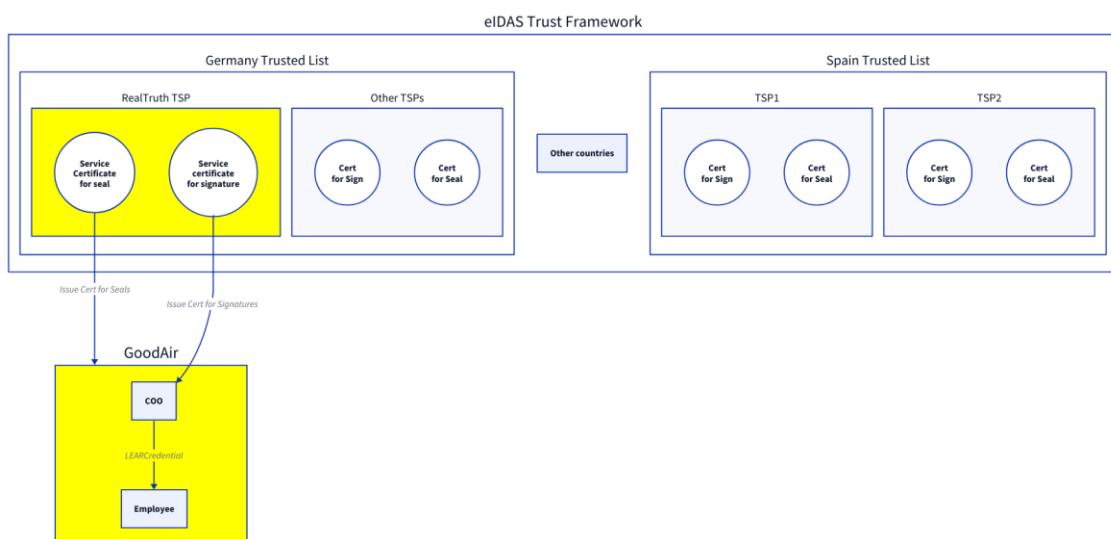
6.1 Participants

Here we introduce the actors of the reference use case we are going to have. The main actors are marked in yellow in the following diagram.

RealTruth is a Trust Service Provider operating under the eIDAS Trust Framework, which issues a certificate for seals to the company GoodAir. GoodAir is a business that provides some services using an Air Quality application operated by GoodAir.

RealTruth also provides a certificate for signatures to the COO of the GoodAir company, so the COO can use the certificate to sign documents on behalf of GoodAir (like contracts, invoices, financial reports, ...) in a way that is compliant with eIDAS.

The COO of GoodAir will issue a verifiable credential of type LEARCredential to an employee of GoodAir, signing the credential with his certificate for signatures.



6.1.1 SmartCityDS: an instance of a Data Space for Smart Cities

SmartCityDS is a Data Space where local governments can procure data and services from other entities that act as service providers. At the same time, the local governments can also act as data and service providers for other entities in the Data Space.

The SmartCityDS Data Space has an onboarding service that allows external entities to perform onboarding in an automated fashion by using Verifiable Credentials that have been issued by trusted entities. The complete Trust Framework used by SmartCityDS is composed from a series of Trust Frameworks, some managed internally using a governance model of SmartCityDS and some others managed externally but by trusted entities. At the root of the Trust Framework of SmartCityDS is the eIDAS Trust Framework and the pan-european recognized list of Trust Service Providers issuing the eIDAS compliant digital identities, in the form of certificates for signatures/seals.

In order to participate in SmartCityDS, every legal person requires a certificate for seals or that one of its legal representatives has a certificate for signatures (or both).

6.1.2 GoodAir: the company that wants to will perform the onboarding process

The new participant is a business providing an Air Quality Monitoring application as a service. The business is called GoodAir and it operates the application, which receives data from a set of sensors that may or may not be the property of GoodAir. The sensors must have received a certification to be able to operate and send data to the GoodAir application.

The company is registered in the tax agency and business registry of Spain with VAT number VATES-12345678

Jesus Ruiz COO (Chief Operating Officer) of GoodAir.

The GoodAir company is small and the only person that can sign contracts on behalf of the company is the COO (Chief Operating Officer). The COO is a legal representative of the company and is registered as so in the business registry of Spain.

The COO has a certificate for electronic signatures issued by one of the TSPs in Spain enabling the COO to perform electronic qualified signatures as legal representative of GoodAir, instead of doing them manually.

The X.509 certificate of the COO has the following contents in the **Subject** field (the example below is derived from a real certificate but with the identifiers modified to be an example):

cn=56565656V Jesus Ruiz

serialNumber=56565656V



```
givenName=Jesus
sn=Ruiz
2.5.4.97=VATES-12345678
o=GoodAir
c=ES
2.5.4.13=Notary:Juan Lopez/Protocol Num:7172/Date:07-06-2021
```

The above set of fields bind the legal identity of the COO with the identity of the business:

- **serialNumber** is the unique number recognized by the Spanish Government for spanish citizens (called NIF).
- **2.5.4.97** is the [official OID for organizationIdentifier](#). The contents of this field in the example certificate is the unique identifier for the legal person GoodAir.

Before issuing a certificate like the above, the TSP has to perform some validations to ensure that in the official source of truth (the business registry of Spain in this case) there is already registered information that states that the COO has indeed powers to act on behalf of GoodAir as a legal representative.

6.1.4 RealTruth: a TSP (Trust Service Provider)

RealTruth is an EU TSP, which appears in the TL (Trusted List) maintained by the [German Government](#).

RealTruth is a TSP based in Germany but operates in most of the countries of the EU and so being able to provide Trust Services across many countries, including Spain.

As all TSPs in the TLs of the Member States, its entry in the TL includes one or more "Services" entries which describe the Trust Services provided by the TSP.

A TSP can have one Service issuing certificates for signatures, another service issuing certificates for seals, another service for timestamping, etc.

The regulator approves or suspends each service from a TSP individually, and the services are the root anchor for a given trust environment.

In our case, the entry for RealTruth in the TL includes a **<TSPService>** entry, and inside it the **<ServiceDigitalIdentity>** entry includes a DER-encoded certificate specifying the digital identity of the root anchor for that trust domain. The certificate for the Service has in the Subject field:

```
2.5.4.97 = VATDE-170173453
CN = DRV QC 11 MA CA 2017ca
```



OU = QC 11 Mitarbeiter CA

O = Deutsche Rentenversicherung Westfalen

C = DE

Which in the **organizationIdentifier** field (OID 2.5.4.97) specifies the unique organisation identifier assigned by the German regulatory authority to the TSP: VATDE-170173453.

RealTruth provides **Legal Person Representative Certificates** which are qualified in accordance with Regulation (EU) No. 910/2014 of the European Parliament and of the Council of 23 July 2014 on electronic identification and trust services for electronic transactions in the internal market.

The certification policies of RealTruth includes the following sentences:

The Registration Authority must verify the following information in order to authenticate the identity of the organisation:

- The data concerning the business name or corporate name of the organisation.
- The data relating to the constitution and legal status of the subscriber.
- **The data concerning the extent and validity of the powers of representation of the applicant.**
- The data concerning the tax identification code of the organisation or equivalent code used in the country to whose legislation the subscriber is subject.

The TSP (RealTruth in this case) performs the verifications against **Authentic Sources**.

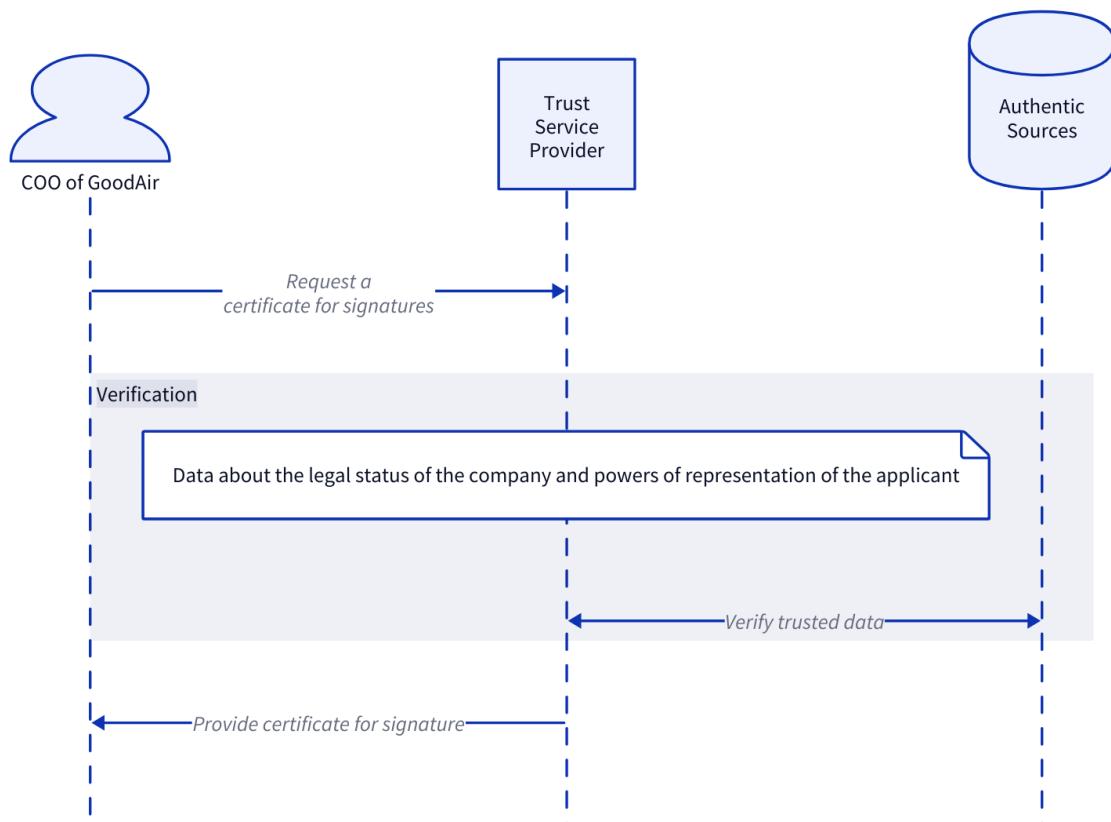
Authentic Sources are the public or private repositories or systems recognised or required by law containing attributes about a natural or legal persons.

The Authentic Sources in scope of Annex VI of the [eIDAS2] legislative proposal are sources for attributes on address, age, gender, civil status, family composition, nationality, education and training qualifications titles and licences, professional qualifications titles and licences, public permits and licences, financial and company data.

Authentic Sources in scope of Annex VI are required to provide interfaces to Qualified Electronic Attestation of Attributes (QEAA) Providers to verify the authenticity of the above attributes, either directly or via designated intermediaries recognised at national level.

Authentic Sources may also issue (Q)EAA-s themselves if they meet the requirements of the eIDAS Regulation. It is up to the Member States to define terms and conditions for the provisioning of these services, but according to the minimum technical specifications, standards, and procedures applicable to the verification procedures for qualified electronic attestations of attributes.





In accordance to the policies, RealTruth made those validations when issuing the certificate to the COO, in such a way that the Relying Parties verifying the certificate can have a high level of trust in the assertion that the natural person identified in the certificate (with Spanish ID of 56565656V) is a legal representative of the GoodAir company (organizationIdentifier VATES-12345678).

6.1.5 John Doe: an administrative employee of GoodAir

This is an employee of the central administration department of GoodAir, who is going to be formally nominated to manage the onboarding process of GoodAir in the Data Space and some other operations in the Data Space once GoodAir is onboarded.. In particular, this employee is going to be nominated as a LEAR (Legal Entity Appointed Representative).

As its name implies, the LEAR has to be nominated by a **legal representative** of the GoodAir organisation with the necessary legal authority to commit the organisation for this type of decision. In our case, the COO of GoodAir will nominate John Doe as the LEAR of GoodAir, delegating to him the capabilities to perform onboarding in the SmartCityDS and some other associated tasks. That means that John Doe will not be empowered to perform other actions like onboarding on other Data Spaces or signing contracts or invoices on behalf of GoodAir.

To perform the nomination, the COO of GoodAir (a legal representative of GoodAir) will issue a special credential to John Doe and will sign the credential with his certificate for signatures as legal representative of GoodAir. The details are described later in this document.

6.2 The LEARCredential

In this example the LEARCredential will be generated using the certificate of the COO.

The credential will be generated with an application that the COO will use as VC Issuer and that allows the employee to receive the credential using his credential wallet, using OIDCVCI to achieve compliance with the EUDI Wallet ARF.

The application enables the COO to specify the information required to create the LEARCredential, specifying the employee information and the specific type of LEARCredential. In general, there may be different instances of LEARCredentials for different purposes. One employee can have more than one LEARCredential, each having a different delegation of powers for different environments.

6.2.1 Claims identifying the employee

These are claims identifying the subject of the credential, the person who will act as LEAR. Each Data Space can define their own depending on their specific requirements. For our example, we use the same that are used for accessing the EC portal. The claims in the credential are the digital equivalent of their analog counterparts, displayed here for illustration.



EU Funding & Tenders: LEAR appointment letter: V6.0 – 01.09.2022

BG CS DA DE EL EN ES ET FI FR GA HR HU IT LT LV MT NL PL PT RO SK SL SV

LEAR APPOINTMENT LETTER

(This document will be automatically generated by the Participant Register once all the information required for the LEAR appointment will have been filled in. You should print it, have it signed by the legal representative and the LEAR and then upload it in the Participant Register with the supporting documents. Originals should be kept on file for controls. If you would like to consult other language versions, please refer to templates & forms section of the [Portal Reference Documents page](#).)

Subject: PIC:
Legal entity name:

I, Mr/Ms/Mrs/Miss, in my capacity as and authorised to legally represent my organisation, have appointed as our legal entity appointed representative (LEAR):

First name:
Last name:
Title: Mr/Ms/Mrs/Miss
Gender:
Postal address (street, postcode, city and country):
e-mail:
Telephone: +(...).
Fax: +(...).
Mobile Phone¹: +(...).

¹ The activation of the LEAR account requires the log in with a PIN code. If you provide a mobile phone number, this PIN code can be sent by SMS. Otherwise we have to send it by post. The number will be used exclusively for sending the PIN code.

Figure 93 - Figure LEAR subject identification data



From the above form we can derive the following claims:

```
{  
    "id": "did:key:xxxxxxxxx",  
    "title": "Mr.",  
    "first_name": "John",  
    "last_name": "Doe",  
    "gender": "M",  
    "postal_address": "",  
    "email": "johndoe@goodair.com",  
    "telephone": "",  
    "fax": "",  
    "mobile_phone": "+34787426623"  
}
```

Where the `id` field is the `did:key` identifier assigned to the employee when issuing the credential, for privacy reasons. This is explained in more detail below.

6.2.2 DID of the employee

In this example we use the `did:key` method for the DID of the employee, which provides a very high level of privacy and given the way the LEARCredential is generated it supports the verification of the chain of responsibility with a proper level of guarantee.

If the highest levels of assurance and legal compliance is desired and the employee has an eIDAS certificate for signatures (in this case a personal one, not one like the COO), we could use the `elsi:did` for identification of the employee. However, the company (GoodAir) should make sure that the employee is aware of and agrees to the possible privacy implications of doing so, given the personal details leaked from the eIDAS certificate.

Those "exposed" personal details are exactly the same as if the employee signs any "normal" document with a digital certificate, but care should be taken by GoodAir because in this case the signature would be done "on behalf of" his employer and not as an individual personal action.

Even though this is not unique to the `did:elsi` method, this also implies that the onboarding service has to handle those personal details in the same way as if it would be accepting any other document signed with a certificate for signatures, and ensure compliance with GDPR.



In this example we assume the usage of the `did:key` method for the employee to protect his privacy as much as possible.

This DID for the employee is an additional claim to the ones presented above, using the `id` field in the `credentialSubject` object. The DID corresponds to a keypair that was generated during the LEARCredential issuance process, where the private key was generated by the wallet of the employee and it was always under his control.

This private key controlled by the employee can be used to sign challenges from Relying Parties that receive the credential to prove that the person sending the credential is the same person that is identified in the `credentialSubject` object of the LEARCredential.

An example could be:

```
{  
  "id": "did:key:99ab5bca41bb45b78d242a46f0157b7d#key1"  
}
```

In this example, the signatures performed with the private key can not be JAdES-compliant ([ETSI-JADES]), but if the LEARCredential is attached to any other credential that is signed with this private key, then they can be traced up to the eIDAS certificate of the COO and so the chain of responsibility can be determined..

With the DID for the employee, the set of claims identifying him would be then:

```
{  
  "id": "did:key:99ab5bca41bb45b78d242a46f0157b7d",  
  "title": "Mr.",  
  "first_name": "John",  
  "last_name": "Doe",  
  "gender": "M",  
  "postal_address": "",  
  "email": "johndoe@goodair.com",  
  "telephone": "",  
  "fax": "",  
  "mobile_phone": "+34787426623"  
}
```



6.2.3 legalRepresentative

This section identifies the natural person (the COO) who is a legal representative of the legal person (GoodAir) and that is nominating the employee identified in the credential.

```
"legalRepresentative": {  
    "cn": "56565656V Jesus Ruiz",  
    "serialNumber": "56565656V",  
    "organizationIdentifier": "VATES-12345678",  
    "o": "GoodAir",  
    "c": "ES"  
}
```

NOTE: Attributes for natural and legal persons

The attributes for natural persons and legal persons are derived from the [eIDAS SAML Attribute Profile \(eIDAS Technical Sub-group, 22 June 2015\)](#).

All attributes for the eIDAS minimum data sets can be derived from the [ISA Core Vocabulary](#) and <https://joinup.ec.europa.eu/collection/semic-support-centre/specifications>.

In the case of natural persons refer to the [Core Person Vocabulary](#) and in the case of legal persons refer to definitions for [Core Business Vocabulary](#).

6.2.4 rolesAndDuties of the LEAR

The **rolesAndDuties** object points to an externally hosted object with the roles and duties of the LEAR. This external object can be either a machine-interpretable definition of the roles and duties in the credential, or just an external definition of the roles and duties in natural language. The ideal approach is the first option, expressing the semantics with a proper machine-readable language, because this will allow automatic access control at the granularity of the individual sentences of that expression language. The **rolesAndDuties** object can also have the definition embedded into it, instead of having a pointer to an external object.

For illustration, the following figure shows an external object with some of the roles and duties in our LEAR example, in natural language.



EU Funding & Tenders: LEAR appointment letter: V6.0 – 01.09.2022

BG CS DA DE EL EN ES ET FI FR GA HR HU IT LT LV MT NL PL PT RO SK SL SV

ROLES AND DUTIES OF LEARS

1. What is a LEAR?

LEAR stands for **legal entity appointed representative**.

For organisations (i.e. not individuals), this is a person formally appointed by the legal representative of the organisation to perform certain tasks on behalf of their organisation, as part of its participation in EU funded grants, procurements and prizes that are managed via the [EU Funding & Tenders Portal](#) — the EU's dedicated website for funding and tenders.

Individuals automatically have the role of LEAR.

2. What can a LEAR do?

As a LEAR you can:

- **view** your organisation's legal and financial data in the Participant Register
- ask to validate **updates of** this information where necessary
- monitor whether or not this information is **validated**, and when
- monitor all uses made of your organisation's **participant identification code** (PIC).

3. What must you do?

As a LEAR you have certain formal obligations:

- **provide** up-to-date legal and financial data (including — on request — supporting documents) on your organisation.
- **maintain and update** this data (*i.e. enabling it to be used for contracting and other transactions between your organisation and the EU*). This means you must **regularly check** that the data is correct and immediately request changes.
- enter and update the names of the colleagues authorised to act as **legal representatives and signatories** for your organisation. These are people who are able to commit your organisation legally by signing grant agreements or contracts and authorising amendments to them.

You must also **revoke** this assignment for any colleague who no longer has these powers.

- enter and update the names of any colleagues authorised to **sign financial statements** or **invoices** on behalf of your organisation.

You must also **revoke** this assignment for any colleague who no longer has this authorisation.

Figure 94 - Figure_ LEAR roles and duties.



A simplistic implementation of the object inside the credential could be:

```
"rolesAndDuties": [
  {
    "type": "LEARCredential",
    "id": "https://www.dome-project.com/lear/v1/6484994n4r9e990494"
  }
]
```

Where the last part of the url can correspond to the hash of the external linked document to ensure that any modification or tampering can be detected.

6.2.5 Assembling the pieces together

With the above values for the example, the complete LEARCredential would become something like this:

```
{
  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://www.dome-
project.com/2022/credentials/learcredential/v1"
  ],
  "id": "urn:did:elsi:25159389-8dd17b796ac0",
  "type": [ "VerifiableCredential", "LEARCredential" ],
  "issuer": {
    "id": "did:elsi:VATES-12345678"
  },
  "issuanceDate": "2022-03-22T14:00:00Z",
  "validFrom": "2022-03-22T14:00:00Z",
  "expirationDate": "2023-03-22T14:00:00Z",
  "credentialSubject": {
    "id": "did:key:99ab5bca41bb45b78d242a46f0157b7d",
```



```
        "title": "Mr.",  
        "first_name": "John",  
        "last_name": "Doe",  
        "gender": "M",  
        "postal_address": "",  
        "email": "johndoe@goodair.com",  
        "telephone": "",  
        "fax": "",  
        "mobile_phone": "+34787426623",  
        "legalRepresentative": {  
            "cn": "56565656V Jesus Ruiz",  
            "serialNumber": "56565656V",  
            "organizationIdentifier": "VATES-12345678",  
            "o": "GoodAir",  
            "c": "ES"  
        },  
        "rolesAndDuties": [  
            {  
                "type": "LEARCredential",  
                "id":  
                "https://www.dome-project.com/lear/v1/6484994n4r9e990494"  
            }  
        ]  
    }  
}
```

6.3 Issuing the LEARCredential

6.3.1 Overview

In this example we use what we call a *profile* of the [OIDCVCI] protocol. The standard is very flexible, and we restrict the different options available in the standard and implement a set of



the options with given values that are adequate for our use case, without impacting flexibility in practice.

The following figure describes the main components that interact in the issuance of a credential in this profile.

The description of the issuance process is general enough to be used for many types of credentials, but the text includes notes describing the concrete application of the process to the case of the LEARCredential.

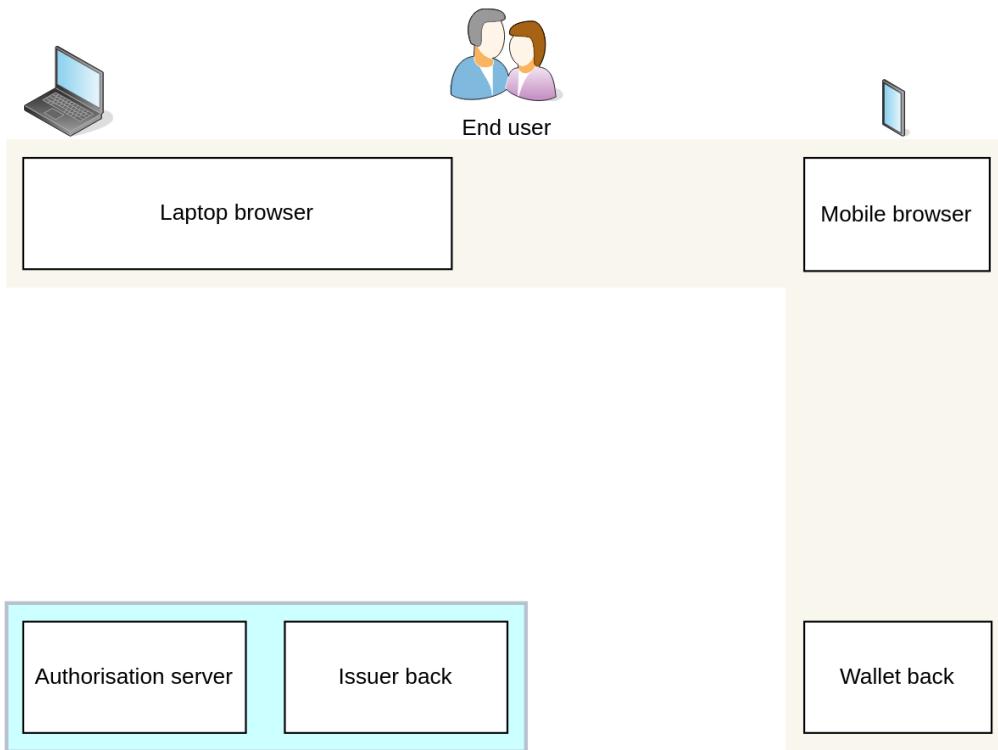


Figure 95 - Figure Overview of participants.

End user

This profile is valid for both natural and juridical persons, but because we are focusing on the issuance of the LEARCredential, in the detailed examples below we assume a natural person as the user.

Wallet

The wallet is assumed to be a Web application with a wallet backend, maybe implemented as a PWA so it has some offline capabilities and can be installed in the device, providing a user experience similar to a native application. Private key management and most sensitive operations are performed in a backend server, operated by an entity trusted by the end user. Other profiles can support native and completely offline PWA mobile applications, for end users.

This type of wallet supports natural persons, juridical persons and natural persons who are legal entity representatives of juridical persons. For juridical persons the wallet is usually called an **enterprise wallet** but we will use here just the term **wallet** unless the distinction is required.

In this profile we assume that the wallet is not previously registered with the Issuer and that the wallet does not expose public endpoints that are called by the Issuer, even if the wallet has a backend server that could implement those endpoints. That makes the wallet implementations in this profile to be very similar in interactions to a full mobile implementation, making migration to a full mobile implementation easier.

In other words, from the point of view of the Issuer, the wallet in this profile is almost indistinguishable from a full mobile wallet.

User Laptop

For clarity of exposition, we assume in this profile that the End User starts the interactions with the Issuer with an internet browser (user agent) in her laptop. However, there is nothing in the interactions which limits those interactions to a laptop form factor and the End User can interact with any internet browser in any device (mobile, tablet, kiosk).

Issuer

In this profile we assume that the Issuer is composed of two components:

- Authorization server: the backend component implementing the existing authentication/authorization functionalities for the Issuer entity.
- Issuer backend: the main server implementing the business logic as a web application and additional backend APIs required for issuance of credentials.

The Issuer backend and the Authorization server could be implemented as a single component in a real use case, but we assume here that they are separated to make the profile more general, especially for big entities and also when using Trust Service Providers for cloud signature and credential issuance, for example.

Authentication of End User and previous Issuer-End User relationship

We assume that the Issuer and End User have a previous relationship and that the Issuer has performed the KYC required by regulation and needed to be able to issue Verifiable Credentials attesting some attributes about the End User. We assume that there is an existing trusted authentication mechanism (not necessarily related to Verifiable Credentials) that the End User employs to access protected resources from the Issuer. For example, the user is an employee or a customer of the Issuer, or the Issuer is a Local Administration and the End User is a citizen living in that city.



6.3.2 Authentication

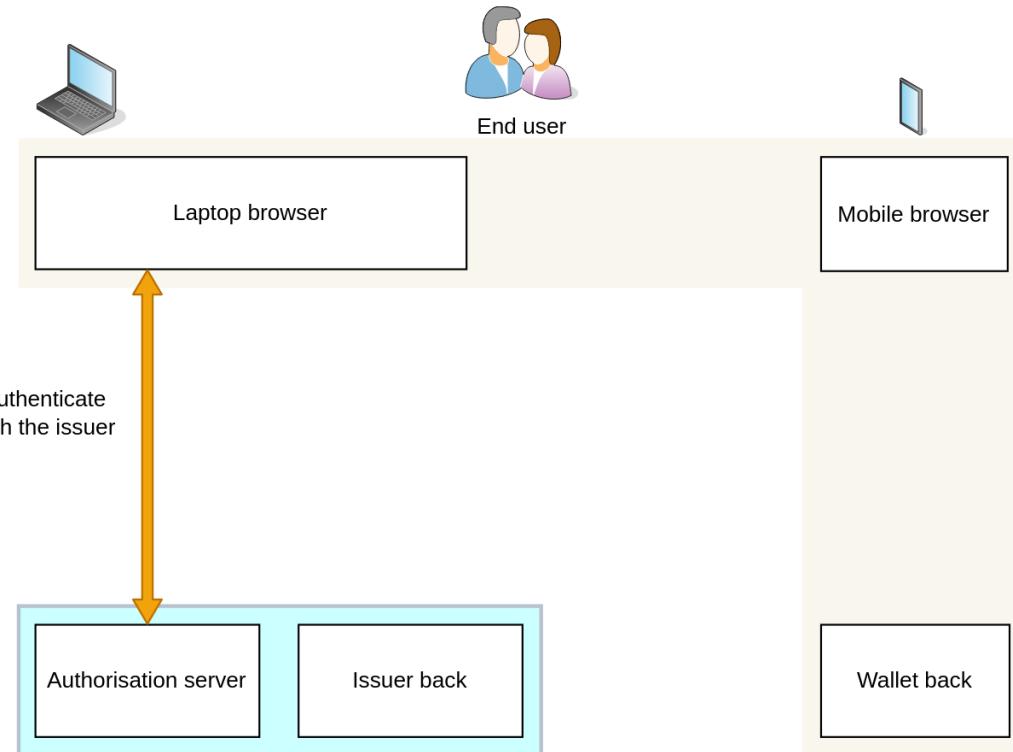


Figure 96 - Figure Issuance authentication.

Before requesting a new credential, the End User has to authenticate with the Issuer with whatever mechanism is already implemented by the Issuer. This profile does not require that it is based on OIDC, Verifiable Credentials or any other specific mechanism.

The level of assurance (LoA) of this authentication mechanism is one of the factors that will determine the confidence that the Verifiers can have on the credentials received by them from a given Issuer.

NOTE: LEARCredential

In the case of the LEARCredential, the End User (John Doe) is an employee of GoodAir and in order to receive the credential John first has to authenticate into the company systems using whatever mechanism GoodAir uses for employee authentication.

Being a modern company, GoodAir uses Verifiable Credentials IAM but this is not a requirement for the issuance of the LEARCredential.

6.3.3 Credential Offer

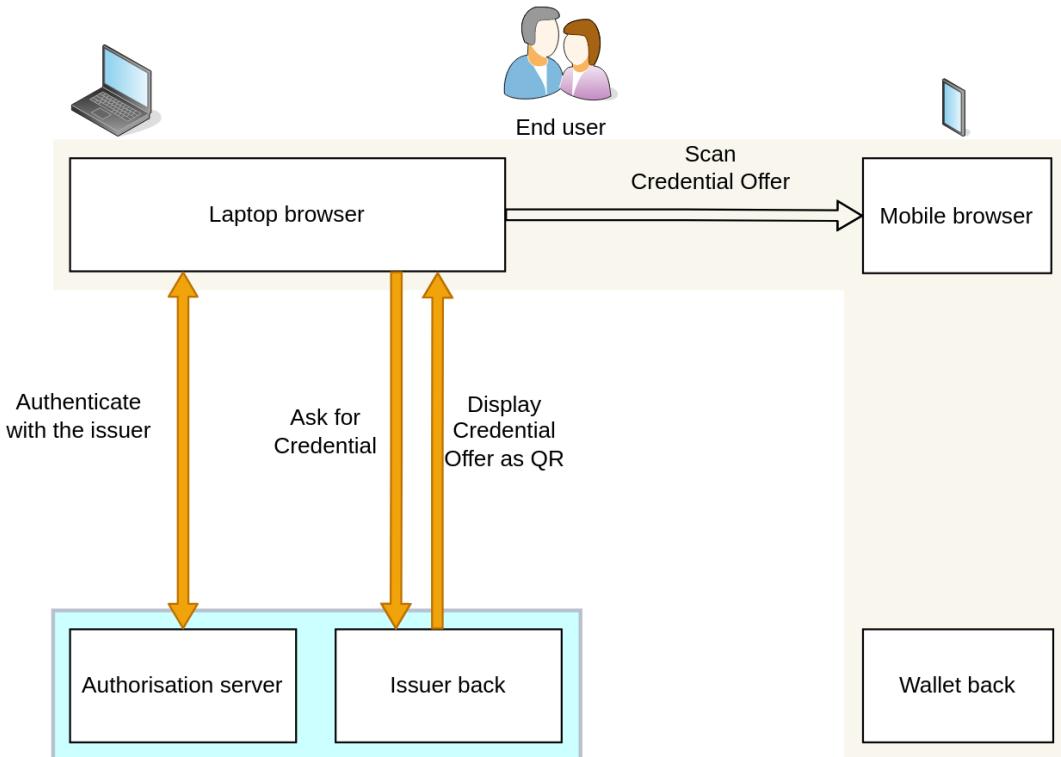


Figure 97 - Figure Credential offer.

In this profile the wallet does not have to implement the Credential Offer Endpoint described in section 4 of [OIDCVCI].

Instead, the Credential Issuer renders a QR code containing a reference to the Credential Offer that can be scanned by the End-User using a Wallet, as described in section 4.1 of [OIDCVCI].

According to the spec, the Credential Offer object is a JSON object containing the Credential Offer parameters and can be sent by value or by reference. To avoid problems with the size of the QR travelling in the URL, this profile requires that the QR contains the `credential_offer_uri`, which is a URL using the `https` scheme referencing a resource containing a JSON object with the Credential Offer parameters. The `credential_offer_uri` endpoint should be implemented by the Issuer backend.

6.3.3.1 Credential Offer Parameters

This profile restricts the options available in section 4.1.1 of [OIDCVCI]. The profile defines a Credential Offer object containing the following parameters:

- **credential_issuer**: REQUIRED. The URL of the Credential Issuer that will be used by the Wallet to obtain one or more Credentials.
- **credentials**: REQUIRED. A JSON array, where every entry is a JSON string. To achieve interoperability faster, this profile defines a global Trusted Credential Schemas List where well-known credential schemas are defined, in addition to the individual credentials that each Issuer can define themselves. The string value *MUST* be one of the id values in one of the objects in the **credentials_supported** metadata parameter of the Trusted Credential Schemas List (described later), or one of the id values in one of the objects in the **credentials_supported** Credential Issuer metadata parameter provided by the Credential Issuer. When processing, the Wallet *MUST* resolve this string value to the respective object. The credentials defined in the global Trusted Credential Schema List have precedence over the ones defined by the Credential Issuer.

NOTE: LEARCredential

The only credential being offered in our case is the LEARCredential, so this is the credential schema that should be specified here. The LEARCredential is a credential known globally to the SmartCityDS ecosystem, so its schema should be published and be available in the Trusted Credential Schemas List.

- **grants**: REQUIRED. A JSON object indicating to the Wallet the Grant Type **pre-authorized_code**. This grant is represented by a key and an object, where the key is **urn:ietf:params:oauth:grant-type:pre-authorized_code**. In this profile the credential issuance flow requires initial authentication of the End User by the Credential Issuer, so the Pre-Authorized Code Flow achieves a good level of security and we do not need the more general Authorization Code Flow.

In other scenarios like when the wallet is a native mobile application and the user interacts with the Issuer exclusively with the mobile (without the laptop), then the Authorization Code Flow has to be used. This can be described in detail in a different profile.

The grant object contains the following values:

- **pre-authorized_code**: REQUIRED. The code representing the Credential Issuer's authorization for the Wallet to obtain Credentials of a certain type. This code *MUST* be short lived and single-use. This parameter value *MUST* be included in the subsequent Token Request with the Pre-Authorized Code Flow.
- **user_pin_required**: REQUIRED. The OIDCVCI standard says it is RECOMMENDED, but this profile specifies the user pin to achieve a greater level of security. This field is a boolean value specifying whether the Credential Issuer expects presentation of a user PIN along with the Token Request in a Pre-Authorized Code Flow. Default is false. This PIN is intended to bind the Pre-Authorized Code to a certain transaction in order to prevent replay of this code by an attacker that, for example, scanned the QR code while standing behind the legit user. It is RECOMMENDED to send a PIN via a separate channel. The PIN value *MUST* be sent in the **user_pin** parameter with the respective Token Request.



The following non-normative example shows a Credential Offer object where the Credential Issuer offers the issuance of one Credential ("LEARCredential"):

```
{  
    "credential_issuer": "https://credential-issuer.example.com",  
    "credentials": [  
        "LEARCredential"  
    ],  
    "grants": {  
        "urn:ietf:params:oauth:grant-type:pre-authorized_code": {  
            "pre-authorized_code": "asju68jgtyk9ikkew",  
            "user_pin_required": true  
        }  
    }  
}
```

6.3.3.2 Contents of the QR code

Below is a non-normative example of the Credential Offer displayed by the Credential Issuer as a QR code when the Credential Offer is passed by reference, as required in this profile:

[https://credential-issuer.example.com/credential-offer?
credential_offer_uri=https%3A%2F%2Fserver%2Eexample%2Ecom%2Fcredential-offer%2F5j349k3e3n23j](https://credential-issuer.example.com/credential-offer?credential_offer_uri=https%3A%2F%2Fserver%2Eexample%2Ecom%2Fcredential-offer%2F5j349k3e3n23j)

Which in plain text would be:

[https://credential-issuer.example.com/credential-offer?
credential_offer_uri=https://server.example.com/credential-offer/5j349k3e3n23j](https://credential-issuer.example.com/credential-offer?credential_offer_uri=https://server.example.com/credential-offer/5j349k3e3n23j)

To increase security, the Issuer *MUST* make sure that every Credential Offer URI is unique for all credential offers created. This is the purpose of the nonce (**5j349k3e3n23j**) at the end of the url in the example. Issuers can implement whatever mechanism they wish, as far as it is transparent to the wallet.



6.3.4 Credential Issuer Metadata

The Wallet backend retrieves the Credential Issuer's configuration using the Credential Issuer Identifier that was received in the Credential Offer.

A Credential Issuer is identified in this context by a case sensitive URL using the https scheme that contains scheme, host and, optionally, port number and path components, but no query or fragment components. No DID is used in this context.

Credential Issuers *MUST* make a JSON document available at the path formed by concatenating the string `/.well-known/openid-credential-issuer` to the Credential Issuer Identifier. If the Credential Issuer value contains a path component, any terminating / *MUST* be removed before appending `/.well-known/openid-credential-issuer`.

`openid-credential-issuer` *MUST* point to a JSON document compliant with this specification and *MUST* be returned using the `application/json` content type.

The retrieval of the Credential Issuer configuration is illustrated below.

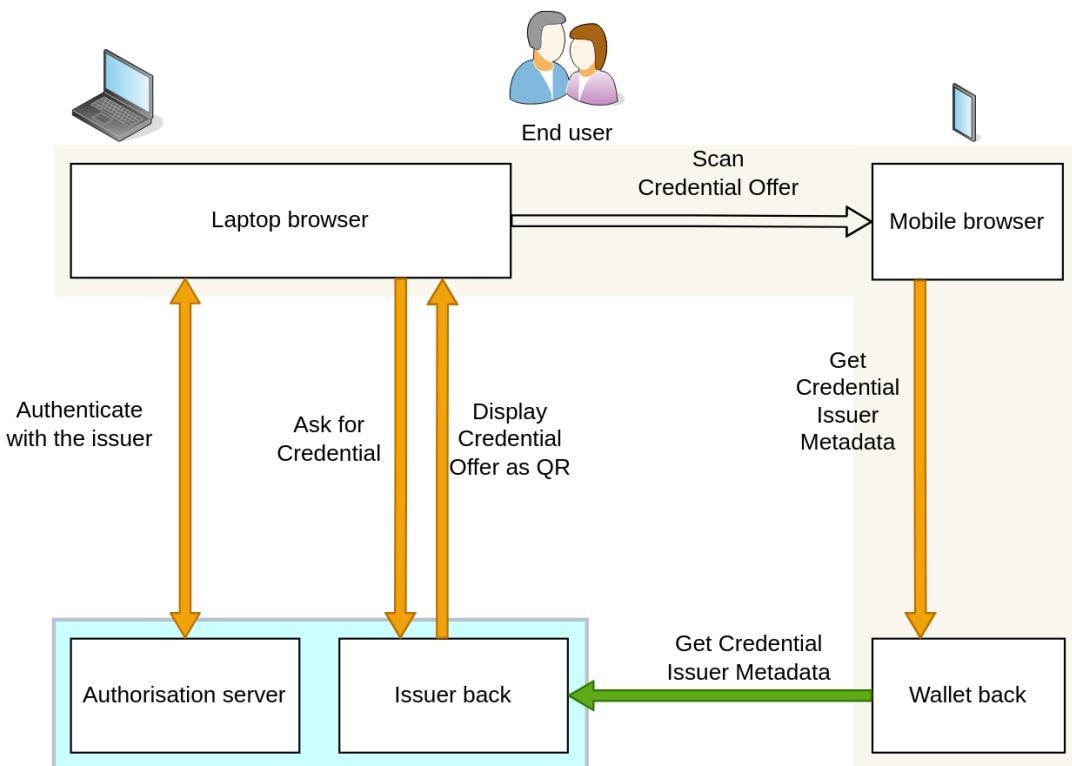


Figure 98 - Figure Issuer metadata.

6.3.4.1 Credential Issuer Metadata Parameters

The object contained in `openid-credential-issuer` contains the following:

- `credential_issuer`: REQUIRED. The Credential Issuer's identifier.
- `credential_endpoint`: REQUIRED. URL of the Credential Issuer's Credential Endpoint. This URL *MUST* use the https scheme and *MAY* contain port, path and query parameter components.
- `credentials_supported`: REQUIRED. A JSON array containing a list of JSON objects, each of them representing metadata about a separate credential type that the Credential Issuer can issue. The JSON objects in the array *MUST* conform to the structure of the Section XXXX.

TODO: define a global directory of credentials supported to eliminate requirement for each individual Issuer to publish its own list.

This profile does not make use of the following parameters:

- `authorization_server` parameter, because it uses the `pre-authorized_code` Grant type.
- `batch_credential_endpoint` parameter. It indicates that the Credential Issuer does not support the Batch Credential Endpoint.
- `display` parameter.

6.3.5 OAuth 2.0 Authorization Server Metadata

This specification also defines a new OAuth 2.0 Authorization Server metadata [RFC8414] parameter to publish whether the AS that the Credential Issuer relies on for authorization, supports anonymous Token Requests with the Pre-authorized Grant Type. It is defined as follows:

- `pre-authorized_grant_anonymous_access_supported`: OPTIONAL. A JSON Boolean indicating whether the issuer accepts a Token Request with a Pre-Authorized Code but without a client id. The default is false.



6.3.6 Access Token

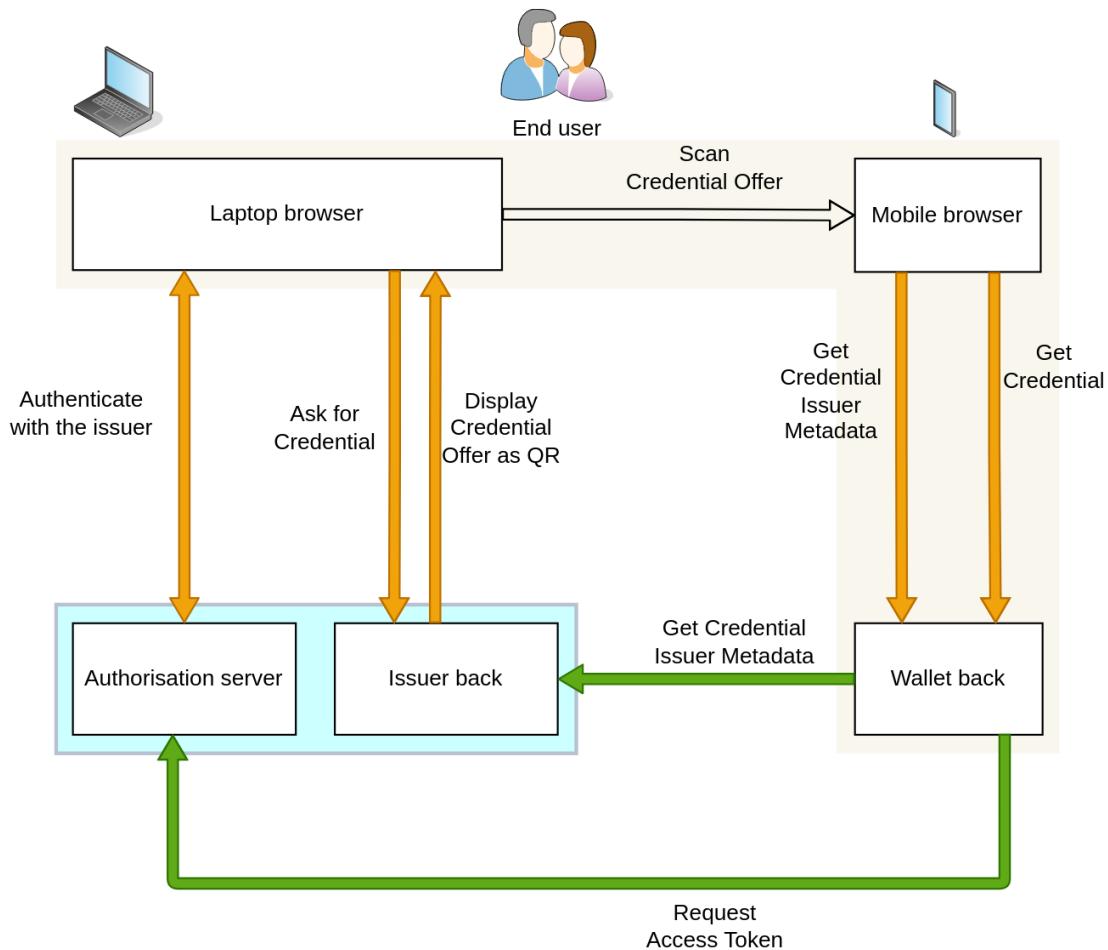


Figure 99 - Figure Access Token.

The Wallet invokes the Token Endpoint implemented by the Authorization Server, which issues an Access Token and, optionally, a Refresh Token in exchange for the Pre-authorized Code that the wallet obtained in the Credential Offer.

6.3.6.1 Token Request

After the wallet receives the Credential Issuer Metadata, a Token Request is made as defined in Section 4.1.3 of [RFC6749].

The following are the extension parameters to the Token Request used in a Pre-Authorized Code Flow as used in this profile:



- **pre-authorized_code**: *REQUIRED*. The code representing the authorization to obtain Credentials of a certain type.
- **user_pin**: *OPTIONAL*. String value containing a user PIN. This value *MUST* be present if user_pin_required was set to true in the Credential Offer. The string value *MUST* consist of maximum 8 numeric characters (the numbers 0 - 9).

In this profile the Wallet does not have to authenticate when using the Token Endpoint, because we are using the Pre-Authorized Code Grant Type, given the level of trust between the Issuer and the End User and that authentication was already performed at the beginning of the flow.

Below is a non-normative example of a Token Request:

```
POST /token HTTP/1.1
Host: server.example.com
Content-Type: application/x-www-form-urlencoded
```

```
grant_type=urn%3Aietf%3Aparams%3Aoauth%3Agrant-type%3Apre-
authorized_code
&pre-authorized_code=Spxl0BeZQQYbYS6WxSbIA
&user_pin=493536
```

6.3.6.2 Successful Token Response

Token Responses are made as defined in [[RFC6749](#)].

In addition to the response parameters defined in [[RFC6749](#)], the Authorization Server returns the following parameters:

- **c_nonce**: *REQUIRED*. JSON string containing a nonce to be used to create a proof of possession of key material when requesting a Credential. The Wallet *MUST* use this nonce value for its subsequent credential requests until the Credential Issuer provides a fresh nonce.
- **c_nonce_expires_in**: *REQUIRED*. JSON integer denoting the lifetime in seconds of the c_nonce.

Below is a non-normative example of a Token Response:

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
```



```
{  
    "access_token": "eyJhbGciOiJSUzI1NiIsInR5cCI6Ikp..sHQ",  
    "token_type": "bearer",  
    "expires_in": 86400,  
    "c_nonce": "tZignsnFbp",  
    "c_nonce_expires_in": 86400  
}
```

6.3.6.3 Token Error Response

If the Token Request is invalid or unauthorized, the Authorization Server constructs the error response as defined in section 6.3 of [OIDCVC1].



6.3.7 Request and receive Credential

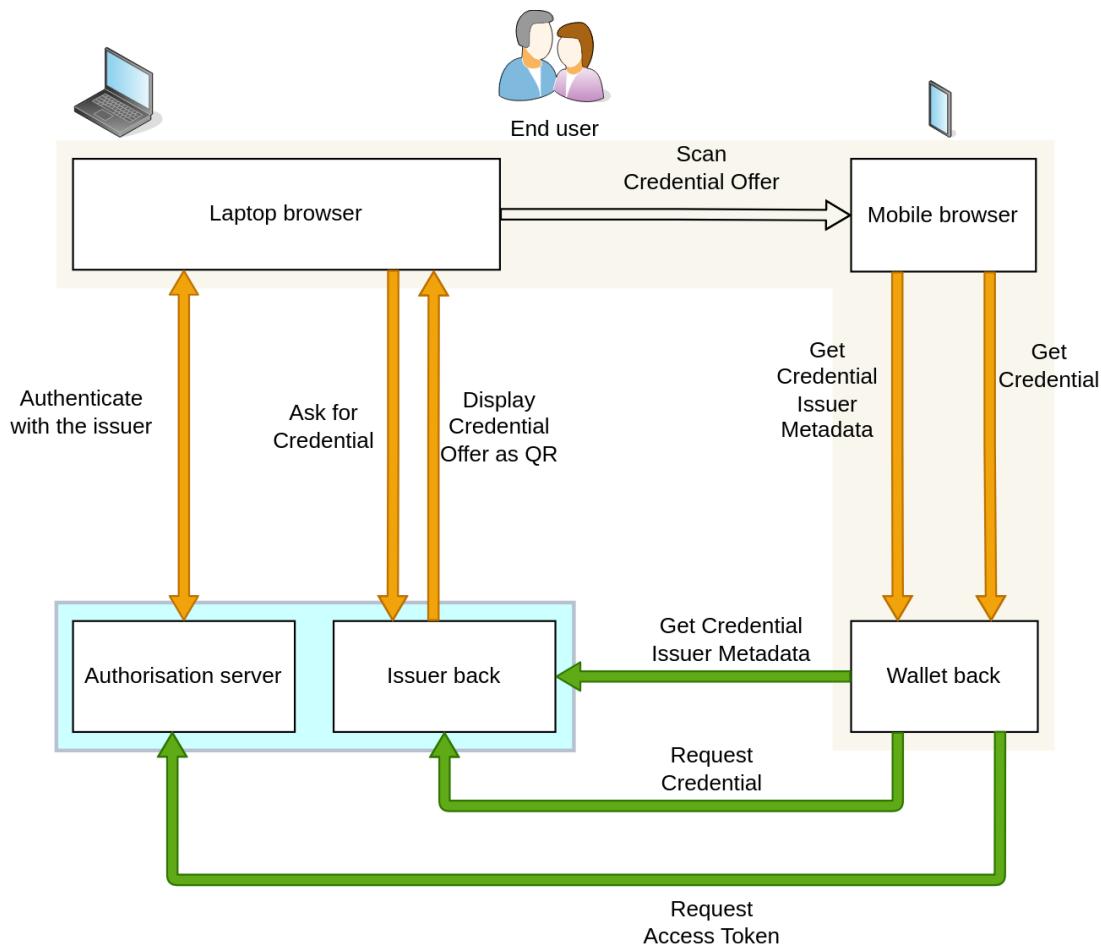


Figure 100 - Figure Request and receive Credential.

The Wallet backend invokes the Credential Endpoint, which issues a Credential as approved by the End-User upon presentation of a valid Access Token representing this approval.

Communication with the Credential Endpoint *MUST* utilize TLS.

The client can request issuance of a Credential of a certain type multiple times, e.g., to associate the Credential with different public keys/Decentralized Identifiers (DIDs) or to refresh a certain Credential.

If the Access Token is valid for requesting issuance of multiple Credentials, it is at the client's discretion to decide the order in which to request issuance of multiple Credentials requested in the Authorization Request.

6.3.7.1 Binding the Issued Credential to the identifier of the End-User possessing that Credential

The Issued Credential *MUST* be cryptographically bound to the identifier of the End-User who possesses the Credential. Cryptographic binding allows the Verifier to verify during the presentation of a Credential that the End-User presenting a Credential is the same End-User to whom that Credential was issued.

The Wallet has to provide proof of control alongside key material (proof that did).

6.3.7.2 Credential Request

The Wallet backend makes a Credential Request to the Credential Endpoint by sending the following parameters in the entity-body of an HTTP POST request using the [application/json](#) media type.

- **format:** *REQUIRED*. This profile uses the Credential format identifier [jwt_vc_json](#).
- **proof:** *OPTIONAL*. JSON object containing proof of possession of the key material the issued Credential shall be bound to. The specification envisions use of different types of proofs for different cryptographic schemes. The proof object *MUST* contain a **proof_type** claim of type JSON string denoting the concrete proof type. This type determines the further claims in the proof object and its respective processing rules. Proof types are defined in Section 7.2.1.

The proof element *MUST* incorporate a **c_nonce** value generated by the Credential Issuer and the Credential Issuer Identifier (audience) to allow the Credential Issuer to detect replay. The way that data is incorporated depends on the proof type. In a JWT, for example, the **c_nonce** is conveyed in the nonce claims whereas the audience is conveyed in the **aud** claim. In a Linked Data proof, for example, the **c_nonce** is included as the **challenge** element in the proof object and the Credential Issuer (the intended audience) is included as the **domain** element.

6.3.7.2.1 Proof Type

This specification defines the following values for **proof_type**:

jwt: objects of this type contain a single **jwt** element with a JWS [RFC7515] as proof of possession. The JWT *MUST* contain the following elements:

- in the JOSE Header,
 - **typ:** *REQUIRED*. *MUST* be [openid4vci-proof+jwt](#), which explicitly types the proof JWT as recommended in Section 3.11 of [RFC8725].
 - **alg:** *REQUIRED*. A digital signature algorithm identifier such as per IANA "JSON Web Signature and Encryption Algorithms" registry. *MUST NOT* be none or an identifier for a symmetric algorithm (MAC).



- **kid**: REQUIRED. JOSE Header containing the key ID. The Credential will be bound to a DID, so the kid refers to a DID URL which identifies a particular key in the DID Document that the Credential will be bound to.
- in the JWT body,
 - **aud**: REQUIRED (string). The value of this claim *MUST* be the Credential Issuer URL of the Credential Issuer.
 - **iat**: REQUIRED (number). The value of this claim *MUST* be the time at which the proof was issued using the syntax defined in [RFC7519].
 - **nonce**: REQUIRED (string). The value type of this claim *MUST* be a string, where the value is the **c_nonce** provided by the Credential Issuer.

The Credential Issuer *MUST* validate that the proof is actually signed by a key identified in the JOSE Header.

Below is a non-normative example of a proof parameter (dots in the middle of jwt for display purposes only):

```
{  
  "proof_type": "jwt",  
  "jwt": "eyJraWQiOiJkaWQ6ZXhhb...aZKPxgiac0aW9EkL1n0zM"  
}
```

where the JWT looks like this:

```
{  
  "typ": "openid4vc1-proof+jwt",  
  "alg": "ES256",  
  "kid": "did:example:ebfeb1f712ebc6f1c276e12ec21/keys/1"  
}  
  
{  
  "iss": "s6BhdRkqt3",  
  "aud": "https://server.example.com",  
  "iat": 1659145924,  
  "nonce": "tZignsnFbp"  
}
```

6.3.7.3 Credential Response



This profile restricts Credential Response to be Synchronous and Deferred response is not used. The Credential Issuer *MUST* be able to immediately issue a requested Credential and send it to the Client.

The following claims are used in the Credential Response:

- **format**: REQUIRED. JSON string denoting the format of the issued Credential. This profile uses the format identifier `jwt_vc_json`.
- **credential**: REQUIRED. Contains issued Credential. *MUST* be a JSON string.
- **c_nonce**: OPTIONAL. JSON string containing a nonce to be used to create a proof of possession of key material when requesting a Credential. When received, the Wallet *MUST* use this nonce value for its subsequent credential requests until the Credential Issuer provides a fresh nonce.
- **c_nonce_expires_in**: OPTIONAL. JSON integer denoting the lifetime in seconds of the c_nonce.

Below is a non-normative example of a Credential Response:

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

{
  "format": "jwt_vc_json",
  "credential" : "LUpixVCWJk0e0t4CXQe1NXK....WZwmhm90Qp6YxX0a2L",
  "c_nonce": "fGFF7UkhLa",
  "c_nonce_expires_in": 86400
}
```

6.3.7.4 Credential Error Response

When the Credential Request is invalid or unauthorized, the Credential Issuer constructs the error response as defined in section 7.3.1 of OIDCVCI.

6.3.7.5 Credential Issuer Provided Nonce

Upon receiving a Credential Request, the Credential Issuer *MUST* require the Wallet to send a proof of possession of the key material it wants a Credential to be bound to. This proof *MUST* incorporate a nonce generated by the Credential Issuer. The Credential Issuer will provide the client with a nonce in an error response to any Credential Request not including such a proof or including an invalid proof.



Below is a non-normative example of a Credential Response with the Credential Issuer requesting a Wallet to provide in a subsequent Credential Request a proof that is bound to a **c_nonce**:

HTTP/1.1 400 Bad Request

Content-Type: application/json

Cache-Control: no-store

```
{  
    "error": "invalid_or_missing_proof",  
    "error_description":  
        "Credential Issuer requires proof to be bound to a Credential  
        Issuer provided nonce.",  
    "c_nonce": "8YE9hCnyV2",  
    "c_nonce_expires_in": 86400  
}
```



7 Appendix II: JAdES signatures of Verifiable Credentials

7.1 Introduction

Verifiable Credentials are typically signed using JWS when using JWT VCs or LD-Proofs when using JSON-LD VCs, the latter being discouraged by the ARF. JWS signatures can be used as per the ARF but since this kind of signatures are not done according to the ETSI Advanced Electronic Signature creation standards those can't be recognized as Advanced Signatures and are therefore considered Simple Signatures. In those cases that an Advanced Signature or a Qualified Signature are required or beneficial, Credentials can be signed using [JAdES](#) or JSON Advanced Electronic Signatures. For more details regarding Electronic Signatures refer to the [3.2.1.1 Legal basis](#) section or the [European eSignature FAQ](#).

Along this section when talking about electronic signatures we are talking about electronic seals too unless otherwise specified.

7.2 Choosing between Qualified and Advanced Signatures

The first step when signing using JAdES is to consider whether to do an Advanced Signature or a Qualified one. Using Qualified Signatures makes the signature of the data legally equivalent to its handwritten counterpart, as said in the European eSignature FAQ “only qualified electronic signatures are explicitly recognized to have the equivalent legal effect of handwritten signatures all over EU Member States.” This means that this kind of signatures provide the highest legal certainty to the involved parties.

For a signature to be considered Qualified it MUST comply with the following:

- It was created by a qualified seal creation device (QSCD). The simplest way would be to use it as a rQSCD or remote QSCD provided by a QTSP.
- Is based on a qualified certificate for electronic seals/signatures.
- Complies with the requisites of previous signature levels, which are those of Simple Electronic Signatures and Advanced electronic signatures (AdES).

Another key point is that the signed data can be time stamped too. The timestamp aims to bind the time stamped data to a particular time establishing evidence that the data existed at that time. Again we have non Qualified and Qualified Timestamps. In this case the advantage of Qualified Timestamps is the presumption of the accuracy of the date and the time it indicates and the integrity of the data to which the date and time are bound. Qualified timestamping MUST be done by a QTSP.



7.3 Signing using JAdES

The simplest way to do a JAdES signature is by using the [European DSS library](#) which covers both the signature creation, validation and timestamping of every ETSI standardised AdES signature types. Other libraries could be used theoretically but this is the recommended approach. The Cloud Signature Consortium is standardising an API to do everything related to the signatures with the same interface independently of the chosen QTSP to provide that remote services. That API and other DSS related information can be read in the section 8 Appendix III: remote Digital Signature Service (rDSS).

7.3.1 Timestamping of JWTs

When signing JWT Credentials only JAdES Baseline-B can be used for compatibility reasons with JWTs and JAdES which means that a JWT can't be time stamped directly. Documentation related to this aspect will be elaborated in the future.

7.3.2 Example of a JWT Credential signed using JAdES

In the following example extracted from EBSI's website we can see a VerifiableID signed using JAdES:

```
{
  "alg": "RS256",
  "cty": "json",
  "kid": "MGcwYKReMFwxCzAJBgNVBAYTA1NJMRQwEgYDVQQKEwtIYWxjb20gZC5kLjEXMBUGA1UEYRMOVkFUU
0ktNDMzNTMxMjYxHjAcBgNVBAMTFUhbgGNvbSBDQSBQTyB1LXN1YlwgMQIDEPSP",
  "x5t#S256": "-zpb6qm4B4NrqhEhjoloohMtoj9jRm7BJXG3jkWB4EQ",
  "x5c": [
    "MIIGYTCCBUmgAwIBAgIDEPSMA0GCSqGSIb3DQEBCwUAMFwxCzAJBgNVBAYTA1NJMRQwEgYDVQQKE
wtIYWxjb20gZC5kLjEXMBUGA1UEYRMOVkFUU0ktNDMzNTMxMjYxHjAcBgNVBAMTFUhbgGNvbSBDQSB
QTyB1LXN1YlwgMTAeFw0x0TEyMjMxMDI4MzNaFw0yMjEyMjMxMDI4MzNaMIH+MQswCQYDVQQGEwJTS
TEMcQGA1UEChMdQU5USE90WSBGSVNRV1gQ0FNSUxMRVJJIFMuUC4xFzAVBgkrBgeEAa4zAgMTCDY
xMDM4NzUwMRcwFQYDVQRhEw5WQVRTSS02MTAzODc1MDEtMCsGA1UEAxMkQW50aG9ueSBGaXNoZXIgQ
2FtaWxsZXJpIFMucC4gRSBTZWFsMQ8wdQYDVQQEEwZFIFN1YlwxJjAkBgNVBCoTHUFudGhvbnkgRml
zaGVyIENhbWlsbGVyaSBTLnAuMS0wKwYJKoZIhvcNAQkBh5hbnRob255QGtub3dsZWRnZWlubm92Y
XRpb24uZXUwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQCq6v/DdquFwMOCDqs3PXRYooE
"
}
```



```

PJ7YXZHvauSL8DncZzCpziPmEDQ9h2NR6sj0M430m5B9nFWXHdXcU18a5wwAuHH9TkKGJIhgSMDDG
8cM6+12Ns6BrnveVJ2L7Wcbi1sTDfoaqZHxe472X3YwhnP7YEZXzt9KgvFO3PyhFEb8y/a3vhL4X30
OTicQJm07GL1LHWVBy28o1z9he3rHFFIN0vH9wHnCzAqbTLcNi0Ync0Jp0Rt1FtZj8FwpdY6RGC18q
AVLaIuG/ASpd6tTd8zs8fyazBOMHMKQ2IM92G+TdnesyER6eMLB7Oj7VKW9I+JEiZaPaCwsBKRAqgn
vvF/DAgMBAAGjggKHMIIICgzATBgNVHSMEDDAKgAhJSHZQdwqxDDCBggYIKwYBBQUHAQMEdjb0MBUGC
CsGAQUFBwsCMAkGBwQAi+xJAQIwCAYGBACORgEBMAgGBgQAjkYBBDAyBgYEAI5GAQUwKDAmFiBodHR
wczovL3d3dy5oYWxjb20uc2kvcmVwb3NpdG9yeRMCRU4wEwYGBACORgEGMAkGBwQAjkYBBgIwgYAGC
CsGAQUFBwEBBHQwcjBNBgrBgfEFBQcwAoZBaHR0cDovL3d3dy5oYWxjb20uc2kvdXBsb2Fkcy9yZXB
vc210b3J5L0hhbGNvbV9DQV9QT191LXN1YWxfMS5jZXIwIQtyIKwYBBQUHMAGGFWh0dHA6Ly9vY3NwL
mhhbGNvbS5zaTBmBgNVSAExzBdMFAGCisGAQQBrjMFAwEwQjBABgrBgfEFBQcCARY0aHR0cDovL3d
3dy5oYWxjb20uc2kvdXBsb2Fkcy9maWxlc9DUFNfaGFsY29tX2NhLnBkZjAJBgcEAIvsQAEDMIGzB
gNVHR8EgaswgagwgaWggaKggZ+GZWxkYXA6Ly9sZGFwLmhhbGNvbS5zaS9jbj1IYwxjb201MjBDQSU
yMFBPJTlZS1zZWf5JTIwMSxvPUhhbGNvbSxjPVNJP2N1cnRpZmljYXR1cmV2b2NhdGlvbmxpc3Q7Y
mluYXJ5hjZodHRwOi8vZG9taW5hLmhhbGNvbS5zaS9jcmxzL2hhbGNvbV9jYV9wb191LXN1YWxfMS5
jcmwwEQYDVR00BAoECEsy60sNP7RzMA4GA1UdDwEB/wQEAvIFoDAYBgYqhXAiAgEEDhMMODg40DAzM
DAwNjc2MAkGA1UdEwQCMAAwDQYJKoZIhvcNAQELBQADggEBAFE+e6vcubeQ4I6Eptx11E2dBxB+DEa
w4m6quPbSZk7yanByp0QRG/rSXFAJC2PDQRVc9k/J096VftrE9tIPyOpEYXXugdLJ5t9ufpkTbGNOp
10/ioxqWcMqvY/vyuXrvsu5wAd0sAmKaruOqNKLSIxoy1xRxZjhFUYIjATK8T6SCVRfojZw81Cx0
TNZHRG79d1EEg5zViy8ZPt41904iCRuzVCUFI1Z81VtAWiEDALWR4VUXAJN5GFLgj6Br26kLxiAB
TLZcYgr8fEPUCU5mNvHWU+gD9yHYv68ploPbEPONK601cTfhvEjPitVOOB+/QBisIr95U3+vkGRsf0
="
    ],
    "typ": "jose",
    "sigT": "2022-04-13T07:18:32Z",
    "crit": [
        "sigT"
    ]
}
.
{
    "iss": "did:ebsi:z219z1CJSbtFc69M2jHcFmq",
    "sub": "did:ebsi:zsSgDXeYPhZ3AuKhTFneDf1",
    "jti": "urn:ebsi:status:identity:verifiableID#1dee355d-0432-4910-ac9c-70d89e8d674e",
    "iat": 1638316800,
    "nbf": 1638316800,
    "exp": 1953849600,
    "vc": {
        "@context": [

```



```
        "https://www.w3.org/2018/credentials/v1"
    ],
    "id": "urn:ebsi:status:identity:verifiableID#1dee355d-0432-4910-ac9c-
70d89e8d674e",
    "type": [
        "VerifiableCredential",
        "VerifiableAttestation",
        "VerifiableId"
    ],
    "issuer": "did:ebsi:z219z1CJKSbtFc69M2jHcFmq",
    "issued": "2021-12-01T12:00:00.0Z",
    "issuanceDate": "2021-12-01T12:00:00.0Z",
    "validFrom": "2021-12-01T12:00:00.0Z",
    "validUntil": "2031-12-01T12:00:00.0Z",
    "expirationDate": "2031-12-01T12:00:00.0Z",
    "credentialSubject": {
        "id": "did:ebsi:zsSgDXeYPhZ3AuKhTFneDf1",
        "familyName": "Doe",
        "firstName": "John",
        "dateOfBirth": "1999-03-22",
        "personalIdentifier": "ES/AT/123456789"
    },
    "credentialSchema": {
        "id": "https://api.preprod.ebsi.eu/trusted-schemas-
registry/v1/schemas/0x14b05b9213dbe7d343ec1fe1d3c8c739a3f3dc5a59bae55eb
38fa0c295124f49#",
        "type": "FullJsonSchemaValidator2021"
    },
    "credentialStatus": {
        "id": "urn:ebsi:status:identity:verifiableID#1dee355d-0432-4910-
ac9c-70d89e8d674e",
        "type": "CredentialStatusList2020"
    },
    "evidence": [
        {
            "type": [
                "DocumentVerification"
            ],
            "verifier": "did:ebsi:z219z1CJKSbtFc69M2jHcFmq",
            "evidenceDocument": [

```



```
        "Passport"
    ],
    "subjectPresence": "Physical",
    "documentPresence": [
        "Physical"
    ]
}
}.<signature>
```



8 Appendix III: remote Digital Signature Service (rDSS)

8.1 Introduction

The DSS open source library implements the standards for Advanced Electronic Signature creation, augmentation and validation. It is aligned with European legislation and eIDAS regulation. The Cloud Signature Consortium (CSC) is a global group of industry, government, and academic organisations committed to drive eIDAS-compliant standardisation of digital signatures in the cloud. The best approach to have an effective cloud signature system for DOME, mainly for JAdES signing and timestamping of Verifiable Credentials but not necessarily limited to it, would be to define an API that extends the already standardised API of the CSC to have a QTSP-independent cloud signature API.

8.2 JAdES cloud signature related endpoints

As explained in the [CSC API v2.0 specification](#) all the CSC endpoints are HTTP POST requests with JSON payloads and JSON responses and therefore they have the HTTP header “Content-Type: application/json”.

A remote signature service as the proposed rDSS MUST use the following style for its base URI to be CSC v2.0 compliant: <https://service.domain.org/xxx/csc/v2/>. In the case of the DOME ecosystem the rDSS service URI provided by DigitelTS will be <https://rdss.digitelts.com/csc/v2/>. All the endpoints MUST use this base URI except the OAuth 2.0 related methods that are necessary for authentication to the rDSS which MAY point to URIs different from the one defined as the base URI. It is important to note that in the future under the new eIDAS 2 it could be possible that this OAuth 2.0 authentication needed to operate the rDSS could be done using OID4VC, that is, Verifiable Credentials. To be standard with the CSC and the actual eIDAS legislation it must be done without using Credentials at this moment.

The remote service SHALL implement a /info method to allow applications to discover the supported API methods and authentication/authorization mechanisms as required by the CSC specification.

There are two steps when obtaining authorization to use the rDSS, service authorization and credential authorization. The latter must not be confused with Verifiable Credentials, it's an independent concept in this case. The former gives access to the rDSS API and the later gives authorization to use the specific keys associated to the client to ensure user control of its signing information when signing.



8.2.1 Service authorization and authentication

The rDSS will use OAuth 2.0 as the service auth/authz mechanism to be future proof since the CSC plans to deprecate the insecure usage of HTTP Basic authentication which is the most simple mechanism accepted as of right now..

8.2.2 Credential authorisation for remote signatures

To be able to remotely sign documents such as Verifiable Credentials additional authentication mechanisms MUST be used for each signing session. Several mechanisms can be used but we will use OAuth 2.0 for the purpose of credential authorisation for the signing session which basically entails checking if the user with access to the signing service is authorised to perform a signature as a certain legal or natural person. In our case the signing session will be just one signing operation each time since it is the most basic case proposed by the CSC.

