ETSI GS PDL 012 V1.2.1 (2023-06)

GROUP SPECIFICATION

Permissioned Distributed Ledger (PDL);
Reference Architecture

Disclaimer

The present document has been produced and approved by the Permissioned Distributed Ledger (PDL) ETSI Industry Specification Group (ISG) and represents the views of those members who participated in this ISG.
It does not necessarily represent the views of the entire ETSI membership.

Reference

RGS/PDL-0012v121 Ref Arch

Keywords

architecture, distributed ledger

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00   Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - APE 7112B
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° w061004871

Important notice

The present document can be downloaded from:
https://www.etsi.org/standards-search

The present document may be made available in electronic versions and/or in print. The content of any electronic and/or print versions of the present document shall not be modified without the prior written authorization of ETSI. In case of any existing or perceived difference in contents between such versions and/or in print, the prevailing version of an ETSI deliverable is the one made publicly available in PDF format at www.etsi.org/deliver.

Users of the present document should be aware that the document may be subject to revision or change of status. Information on the current status of this and other ETSI documents is available at https://portal.etsi.org/TB/ETSIDeliverableStatus.aspx

If you find errors in the present document, please send your comment to one of the following services:
https://portal.etsi.org/People/CommiteeSupportStaff.aspx

If you find a security vulnerability in the present document, please report it through our Coordinated Vulnerability Disclosure Program:
https://www.etsi.org/standards/coordinated-vulnerability-disclosure

Notice of disclaimer & limitation of liability

The information provided in the present deliverable is directed solely to professionals who have the appropriate degree of experience to understand and interpret its content in accordance with generally accepted engineering or other professional standard and applicable regulations.
No recommendation as to products and services or vendors is made or should be implied.
No representation or warranty is made that this deliverable is technically accurate or sufficient or conforms to any law and/or governmental rule and/or regulation and further, no representation or warranty is made of merchantability or fitness for any particular purpose or against infringement of intellectual property rights.
In no event shall ETSI be held liable for loss of profits or any other incidental or consequential damages.

Any software contained in this deliverable is provided "AS IS" with no warranties, express or implied, including but not limited to, the warranties of merchantability, fitness for a particular purpose and non-infringement of intellectual property rights and ETSI shall not be held liable in any event for any damages whatsoever (including, without limitation, damages for loss of profits, business interruption, loss of information, or any other pecuniary loss) arising out of or related to the use of or inability to use the software.

Copyright Notification

Contents

List of figures

List of tables

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The declarations pertaining to these essential IPRs, if any, are publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (https://ipr.etsi.org/).

Pursuant to the ETSI Directives including the ETSI IPR Policy, no investigation regarding the essentiality of IPRs, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

DECT™, **PLUGTESTS™**, **UMTS™** and the ETSI logo are trademarks of ETSI registered for the benefit of its Members. **3GPP™** and **LTE™** are trademarks of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners. **oneM2M™** logo is a trademark of ETSI registered for the benefit of its Members and of the oneM2M Partners. **GSM**® and the GSM logo are trademarks registered and owned by the GSM Association.

Foreword

This Group Specification (GS) has been produced by ETSI Industry Specification Group (ISG) Permissioned Distributed Ledger (PDL).

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

Executive summary

The present document is a revision of ETSI GS PDL 012 (V1.1.1) that defines an ETSI-ISG-PDL Reference Architecture (RA) for a permissioned Distributed Ledger Technology (PDL) platform. The present document also describes the characteristics and behaviour of such a platform, along with the services that it can provide and solutions that can be built using it. It includes improved definitions, additional clauses and tables resulting from feedback received from stakeholders using the previous version since its release.

Introduction

The present document defines an ETSI-ISG-PDL Reference Architecture (RA) for a permissioned Distributed Ledger Technology (PDL) platform. The present document also describes the characteristics and behaviour of such a platform, along with the services that it can provide and solutions that can be built using it.

The ETSI-ISG-PDL RA is a template for defining a solution to a particular problem domain (in this case, a PDL platform). The ETSI-ISG-PDL RA describes abstract functional components that support specific sets of functionalities and reference points that describe standard interactions between different parts of the platform and users of that platform (refer to Figure1). The present document uses a Functional Block architecture to define three key aspects of a PDL Platform:

- Standardized platform services, which are services and functionality provided by the PDL platform that conform with pre-defined requirements so they can interoperate with other components of the platform.
- Abstraction layers, which are Data Model Brokers allowing different and diverse applications on one side and different PDL chain types on the other side to interface with the PDL platform.
- Modularity, which allows evolution and adaptation of PDL platforms to changing requirements.

The objectives of using the RA are to:

- Maximize the choice of technology solutions available to entities using ETSI-ISG-PDL-endorsed technologies, Services, and applications.
- Maximize ETSI-ISG-PDL endorsed PDL platforms' scalability in terms of the applications supported and the number of entities able to use them.

The ETSI-ISG-PDL RA also provides standardized terminology to simplify the interaction between objects such as PDL Platforms, Services, and applications (as defined in clause 3.1) developed by ETSI-ISG-PDL members including operators and technology vendors/developers.

This approach enables operators and technology vendors/developers to focus on their respective areas of expertise and market leadership by providing solutions for one or more Reference Architecture functional components and/or services. It also allows users to choose appropriate vendors and solutions for their specific environment and product portfolio.

The architecture aims to be independent of specific implementations to accommodate a wide range of technology solutions that comply with both the requirements of the supported applications and ensures adherence to critical architectural requirements such as interoperability, security, privacy, etc.

The ETSI-ISG-PDL RA comprises two categories of architectural components - those components mandated in all platforms (i.e. PDL Mandatory Platform Services), and those components that are optional and may be included or excluded depending on the applications implemented on the Platform (i.e. PDL Optional Platform Services). This approach facilitates the introduction and support of new applications in a structured manner without changing the common, mandatory, parts. The RA also supports the concept of a distributed lifecycle for applications, where different parties take different roles and responsibilities (for example Buyer versus Seller). This expands the vendor-operator space, by allowing vendors to focus on, and operators to choose from, specific architectural components in the stacks and focus their offerings on the different PDL Platforms, Services, and applications.

Intended Audience

All ETSI members, any technical, commercial and operations experts working in the ICT industry, software vendors, standards organizations, other service providers, and industry bodies.

1 Scope

1.1 Definition

The present document defines a RA for a Permissioned Distributed Ledger platform. Following the terminology and general architectural requirements, the present document discusses the architectural components listed below:

a) Orchestration, governance, process management, and eco-system coordination in a complex environment (for example node and PDL management in an environment involving multiple competing parties or supply chains).

b) External and internal information exchange (for example through Oracles, APIs, micro services, webhooks, or external data sources).

c) Off-chain Storage (another chain, local/cloud node that is not part of the PDL, PDL node but not sharing with other nodes of said PDL, trusted by a single node or trusted by all nodes based on governance, etc.).

d) Smart Contracts (including commonalities between Smart Contracts, interoperability of Smart Contracts across chains, PDL agnosticism).

1.2 In scope

a) Definition of functionalities, interfaces, reference points (for example Identity services such as PDL identity, Node identity, and User identity).

b) Functional capabilities of commercial applications using PDL to create/trade value (for example wholesale settlement, cryptocurrency or stable coins-based payments, tokenization, and asset/inventory management).

c) Non-functional capabilities of commercial applications using PDL to create/trade value (for example network design, security, privacy, and access control).

d) Capabilities of different PDL protocols (common aspects of PDL protocols that can be use case, platform, application, and service agnostic amplified through inter-ledger interoperability).

1.3 Out of scope

a) Architecture design and implementation details.

b) PDL network design and implementation details.

c) Design and implementation details of platforms built on PDL network.

d) Specific application/ service implementation details of application platforms built on PDL network (for example implementation of identity using a specific method).

NOTE: Any platform, application, or network specific implementation details will be added at a later phase through dedicated ETSI-ISG-PDL PRD documents (Including Functional, Privacy and Security related services and requirements) or an annex to the present document (for other services) on need basis.

2 References

2.1 Normative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

Referenced documents which are not found to be publicly available in the expected location might be found at https://docbox.etsi.org/Reference.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are necessary for the application of the present document.

Not applicable.

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] MEF 55.1: "Lifecycle Service Orchestration (LSO): Reference Architecture and Framework", January 2021.

[i.2] NIST Special Publication 800-162, January 2014: "Guide to Attribute Based Access Control (ABAC) Definition and Considerations".

[i.3] Gamma E., Helm R., Johnson R., Vlissides J.: "Design Patterns: -Elements of Reusable Object-Oriented Software", Addison-Wesley, Nov 1994.-ISBN 978-0201633610.

3 Definition of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the following terms apply:

abstraction layer: functionality that serves as an intermediator between subsystems that may be using different protocols, vocabulary, and methods that serves their respective purposes

access control policy: privileges and permissions of a subject entity to perform operations on a set of target entities

addressable storage: content/data that can be accessed through a web link (URL)

API Broker: software that mediates between two systems with different Data Models implemented as APIs

NOTE: Also referred to as API Gateway.

API Gateway: See API Broker.

application: software, program or group of programs designed to perform specific tasks for end users

application abstraction layer: APIs and interfaces, including API Brokers, enabling applications to communicate with a Platform

Application Programming Interface (API): system of tools and resources in an operating system, enabling developers to create software applications

asynchronized data: data that does not require synchronization with other data

Attribute Based Access Control (ABAC): access control method where the subject requests for performing an operation on objects are granted/denied based on:

- Assigned attributes of the subject.
- Assigned attribute of the object.
- Environmental conditions.
- Set of policies.

blockchain: censorship and tamper-proof growing list of records, called blocks, that are linked using cryptography

NOTE: Each block contains a cryptographic hash of the previous block, a timestamp, and transaction data.

business service: service delivered to business customers by business units

category alpha application: application that is developed and delivered to all users of said application by a single vendor/developer using a Category Alpha Platform developed by that same vendor/developer

NOTE: Can only use a single PDL type prescribed by the developer.

category alpha platform: PDL platform that is designed, developed, delivered, and integrated to all users of said platform by a single vendor using a single PDL technology

NOTE: Broken down to sub-categories "Alpha-1" and "Alpha-2".

category bravo application: application developed and delivered to all users of said application by a single vendor/developer using a Category Bravo Platform developed by that same vendor/developer

NOTE: Can only use PDL types prescribed by the developer.

category bravo platform: PDL platform designed, developed, delivered, and integrated to all users of said platform by a single vendor, but can operate using two or more underlying PDL technologies

NOTE: Broken down to sub-categories "Bravo-1" and "Bravo-2".

category charlie application: application developed towards a specification of an Application so that any user of an application supporting such specifications can fully interoperate with other users of other applications built towards the same Application specifications

category charlie platform: PDL platform that can operate using two or more underlying PDL technologies and is designed and developed towards a specification of an Application Abstraction Layer so that any Application that supports such an abstraction layer can interface with said platform

NOTE: Broken down to sub-categories "Charlie-1", "Charlie-2", "Charlie-3" and "Charlie-4".

category delta platform: category charlie platform that only supports a single PDL type

NOTE: Broken down to sub-categories "Delta-1", "Delta-2", "Delta-3" and "Delta-4".

Certificate Authority (CA): entity that issues digital certificates

NOTE: A digital certificate certifies the ownership of a public key by the named subject of the certificate.

composite application: application using the PDL platform that are made up of other applications that use the PDL platform

composition: act of creating a new object or a new functionality through combination of two or more existing objects or functionalities

concurrency: occurrence of and/or execution at the same time of different programmatic units

consumer: PDL Platform entity that consumes data produced by another entity

data model: concepts of interest to an environment in a form that is dependent on data repository, data definition language, query language, implementation language, and/or protocol

NOTE: Data Models are derived from the Information Model.

data model broker: software that mediates between two systems with different data models

NOTE: also referred to as data model gateway.

data model gateway: same as data model broker

directly connected storage: storage that is local to the node and is either physically connected to the node or is external storage connected using a shared communication channel that is managed by the owner of that node

NOTE: Examples of physically connected storage: internal drive, external thunderbolt drive. Examples of external storage: NAS, Cloud.

Discretionary Access Control (DAC): Access Control Policy where the owner of a resource/object defines the Access Control Policy for the users

distributed addressable storage: Addressable Storage that is distributed across multiple storage devices

Distributed Ledger Technology (PDL): technology implementing a distributed ledger which is a consensus of replicated, shared, and synchronized digital data geographically spread across multiple sites, countries, or institutions

NOTE: Unlike with a distributed database, there is no central administrator.

Domain Name System (DNS): hierarchical and decentralized naming system for computers, services, or other resources connected to the Internet or a private network

NOTE: It associates various information with domain names assigned to each of the participating entities.

external data: data obtained from resources or systems external to the PDL platform

external IRP: IRP used to communicate between a PDL Functional Block and an external object

fork: split of a PDL chain into two chains that share the history up to the point where the fork occurred, and then each part is headed in its own direction

functional block: abstraction that defines the external structural representation of the capabilities and functionality of a component or module, and its relationships with other Functional Blocks

NOTE: Functionalities such as capabilities, behaviour, and relationships, as well as their inputs, outputs, and optionally, transfer functions. The internal structure of a Functional Block is not revealed.

functional capability: capability of a system to manage resource in each functional area of operations

governance: collection of rules and tools that control the behaviour and function of a PDL platform

hardware interface: point across which electrical, mechanical, and/or optical signals are conveyed from a sender to one or more receivers using one or more protocols

implementation agreement: rules and agreements that describe how a Platform Service is implemented

information model: representation of concepts of interest to an environment in a form that is independent of data repository, data definition language, query language, implementation language, and protocol

insignificant event: event that does not affect any node other than the node where it occurred and does not affect the chain or consensus mechanism

Interface Reference Point (IRP): communication channels through which Functional Blocks communicate with each other

NOTE: IRPs are given names for reference purposes (for example "Debka").

internal data: data generated by a node either through computation or through a directly connected sensor that feeds data to that node

internal IRP: IRP used to communicate between two or more PDL Functional Blocks

NOTE: This communication stays within the PDL Platform and is not seen by objects that are external to the PDL

Internet Corporation for Assigned Names and Numbers (ICANN): American multi-stakeholder group and non-profit organization responsible for coordinating the maintenance and procedures of several databases related to the namespaces and numerical spaces of the Internet, ensuring the network's stable and secure operation

Internet Engineering Task Force (IETF): open standards organization, which develops and promotes voluntary Internet standards, in particular the standards that comprise the Internet protocol suite

InterPlanetary File System (IPFS): protocol and peer-to-peer network for storing and sharing data in a distributed file system that uses content-addressing to uniquely identify each file in a global namespace connecting all computing devices

loosely coupled: functionality that has little or no dependency on other functionalities

Mandatory Access Control (MAC): Access Control Policy defined by system administrators

Minimum Viable Product (MVP): version of a product with just enough features to satisfy early customers and provide feedback for future product development

network: In the context of the present document, technical infrastructure that allows applications to access PDLs through use of Platform Services.

NOTE: This term is interchangeable with Platform in the context of the present document.

non-addressable storage: content/data that cannot be addressed and accessed by any other entity except for the entity that directly manages this data

orchestration: automated (and/or manual) configuration and management of systems and their Functional Blocks

NOTE: Orchestrated objects may be Resources, Platform Services, Applications. Orchestration emphasizes coordinated actions; one form of this coordination is service function chaining.

PDL abstraction layer: APIs and interfaces, including API Brokers, enabling Platform services to communicate with ETSI-ISG-PDL endorsed PDL types

platform: in the context of the present document, network environment in which one or more applications and services are implemented and executed

platform atomic service: platform service that does not use any other Platform Service to perform its functionality

NOTE: May use external applications or functions.

platform composite service: platform service that uses one or more other platform services to perform its functionality

platform mandatory service: platform service mandated to be included in a platform

platform optional service: platform Service that does not need to be included in a platform for it to be considered ETSI-ISG-PDL compliant

platform service: service implemented within the Platform Services layer that is compliant with the ETSI-ISG-PDL requirements and definitions

policy: set of rules used to manage and control the changing and/or maintaining of the state of one or more managed objects, defined by the Governance

Policy Based Access Control (PBAC): Access Control method that uses Policies to determine the appropriate type of access control based on the needs of the PDL Platform

producer: PDL Platform entity that generates data that other entities may consume

RAM swap space: portion of a computing device's hard drive used for virtual memory in the event that there is insufficient physical RAM installed on the device

Random Access Memory (RAM): hardware in a computing device where the operating system, application programs and data in current use are kept so they can be quickly reached by the device's processor

Reference Architecture (RA): template for defining a solution to a particular problem domain

Remote Procedure Call (RPC): computer program that causes a procedure to execute in a different address space, which is coded as if it were a normal procedure call, without the programmer explicitly coding the details for the remote interaction

Role Based Access Control (RBAC): access control approach based on the roles the user assumes in a system, rather than the user's identity

service: instance of a technology product implemented using a platform

NOTE: For example, a communication circuit connection between two offices.

significant event: event occurred on any node that may affect the behaviour of the node, the chain, or the consensus mechanism

smart contract: program stored on a Blockchain that executes when predetermined conditions are met

software interface: point through which communication with a set of resources of a set of objects is performed

NOTE: Resources such as memory, CPU, Location, User roles or Smart Contracts.

software reference model: set of architectural patterns and other supporting artifacts that presents a set of unifying terminology, concepts, axioms, and Functional Blocks within a particular problem domain

synchronized data: data that requires sequencing and has dependency on timing or content of other data being collected

tightly coupled: functionality that has a high degree of dependency on other functionalities

trusted third parties: in cryptography, entity which facilitates interactions between two parties who both trust the third party

NOTE: The Third Party reviews all critical transaction communications between the parties, based on the ease of creating fraudulent digital content.

Universal Resource Locator (URL): reference to a web resource that specifies its location on a computer network and a mechanism for retrieving it

NOTE: A URL is a specific type of Uniform Resource Identifier, although many people use the two terms interchangeably.

use case: specific situation in which a product or service could potentially be used

virtual service: service that uses one or more virtual objects

NOTE: Objects such as Resources, Services.

3.2 Symbols

Void.

3.3 Abbreviations

For the purposes of the present document, the following abbreviations apply:

ABAC	Attribute Based Access Control
API	Application Programming Interface
CA	Certificate Authority
CPU	Central Processing Unit
DAC	Discretionary Access Control
DNS	Domain Name System
DSL	Domain Specific Language
ETSI	European Telecommunications Standards Institute
FCAPS	Fault, Configuration, Accounting, Performance, Security
GDPR	General Data Protection Regulation
GPS	Global Positioning System
GUI	Graphical User Interface
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
IETF	Internet Engineering Task Force
IP	Internet Protocol
IPFS	InterPlanetary File System
IRP	Interface Reference Point
ISG	Industry Specification Group
ISO	International Organization for Standardization
LSO	Lifecycle Service Orchestration
MAC	Mandatory Access Control
MEF	MEF Forum, formerly known as Metro Ethernet Forum
NAS	Network Attached Storage
PBAC	Policy Based Access Control
PC	Personal Computer
PDL	Permissioned Distributed Ledger
DLT	Distributed Ledger Technology
RA	Reference Architecture
RAM	Random Access Memory
RBAC	Role Based Access Control
SASE	Secure Access Service Edge
SDO	Standards Defining Organization
SFTP	Secure File Transfer Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
TMS	Transaction Management Service
URL	Universal Resource Locator

4 ETSI-ISG-PDL Reference Architecture

4.1 Introduction

A software reference architecture provides a template for defining standardized solutions to a particular problem domain (for example an interoperable settlement platform) by reducing the interoperability overheads in accordance with applicable business rules, regulations, and other constraints. Thus, a software reference architecture specifies the salient characteristics and behaviour of a platform. This takes the form of a set of functions and services that can be used to build more complex and detailed functions and services.

4.2 Definition of a Functional Block

The present document uses a Functional Block architecture to define a software reference architecture. A Functional Block is an abstract concept that defines a "black box" structural representation of the functionality (i.e. capabilities, behaviour and relationships) of a component, module, or system. A software reference model is an abstract definition of a set of architectural patterns and other supporting artifacts that presents a set of unifying terminology, concepts, axioms, and Functional Blocks within a particular problem domain. A set of Functional Blocks interact using a set of Internal and External IRPs (Interface Reference Points) that standardize communication, and collectively define the functionality provided independent of specific technologies, implementations, or other concrete details.

4.3 Definition of an Interface Reference Points (IRP)

An IRP is a logical point of interaction which defines communication channels through which the Functional Blocks defined above communicate with each other.

4.4 Reference Architecture Overview

4.4.1 Conceptual layered architecture

The ETSI-ISG-PDL RA depicted in Figure 1 is a software reference architecture for ETSI-ISG-PDL PDL platforms.

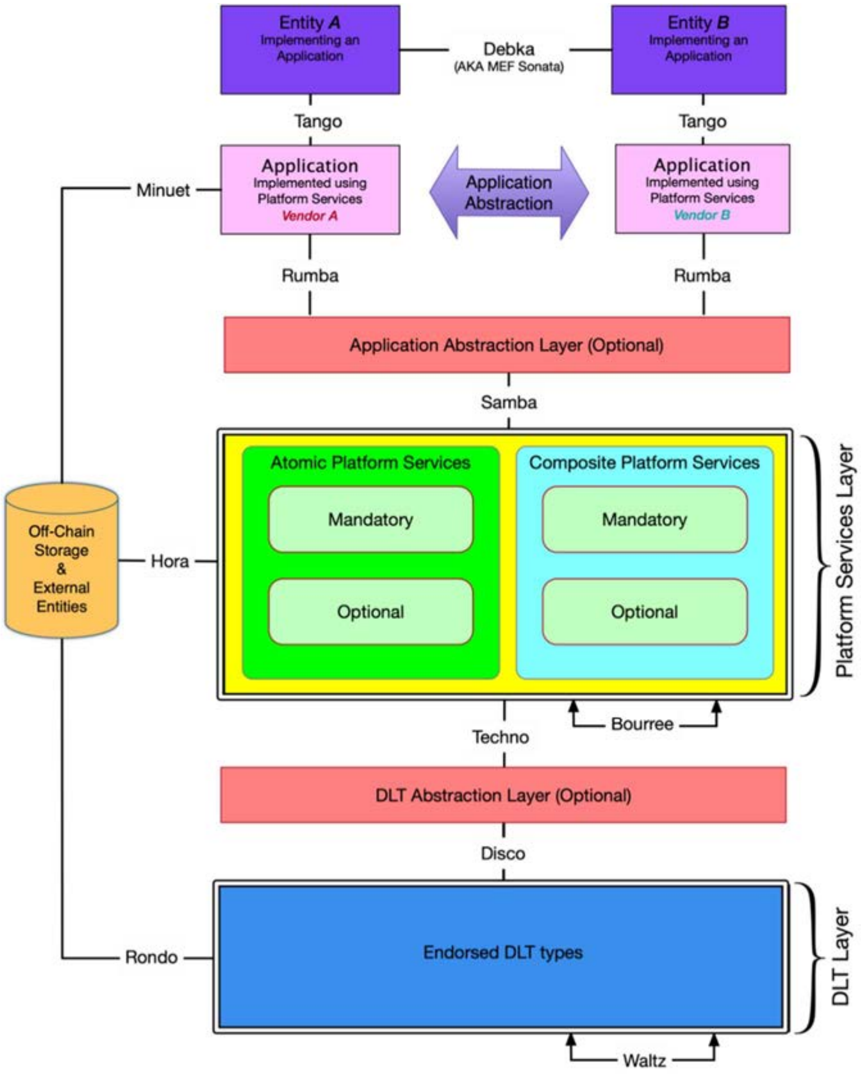

Figure 1: ETSI-ISG-PDL Reference Architecture

The ETSI-ISG-PDL RA is a modular architecture, and reuses individual Functional Blocks to compose new, more powerful, Functional Blocks. In addition, IRPs are defined between Functional Blocks accordingly.

4.4.2 User layer

The User layer, depicted as "Entity A" and "Entity B", represents the users of the platform. Such users could be individuals, companies, other software or any other entity using the PDL platform through a PDL application. This layer is optional in in certain cases may be combined into the Applications layer.

4.4.3 Applications Layer

Consists of Applications using PDL technology.

4.4.4 Application Abstraction Layer

Data Model Brokers/Gateways enabling Applications that use different data models to communicate with platforms. This layer is located between the "Samba" and "Rumba" IRPs and implemented through the Data Model Broker Platform Service where necessary. This layer is optional and may be omitted in clauses 4.5.1.2 and 4.5.1.3 platform types as defined in clause 4.5.1 below.

4.4.5 Platform Services Layer

4.4.5.1 Introduction

The Platform Services Layer supports various types of applications. The Platform Service Layer can provide useful services for applications. As a result, an application could simply leverage services from the PDL Service Layer, which will reduce the application's complexity, accelerate application development and deployment, and increase interoperability. This is where the Application Abstraction Layer is essential, as it enables the continuing development of the PDL architecture to proceed independently of any specific requirements of interacting with external entities. For example, the PDL Service Layer could have a Transaction Management Service to facilitate an application to easily create transactions without knowing details of a specific PDL type (i.e. a specific deployed PDL network); in essence, this Transaction Management Service can perform transaction transformation/adaptation between applications running on different PDL types to facilitate application operations in a complex environment. For abstraction purposes the Service Layer is divided into sub-groups according to the matrix defined in Table 1 herewith. Applications' access to services is independent of service classification and is subject to governance, identity, privacy, and security considerations.

Table 1: Types of Platform Services

	Mandatory	Optional
Atomic	Mandatory Platform Atomic Services	Optional Platform Atomic Services
Composite	Mandatory Platform Composite Services	Optional Platform Composite Services

Certain services and applications do not require a PDL platform to operate. Implementation on a PDL platform provides capabilities such as a trusted source of truth between untrusting parties, immutability of records and disintermediation of third parties. Such capabilities may offer added value in certain contexts.

4.4.5.2 Mandatory PDL Platform Services

Services that a PDL platform has to include to be considered ETSI-ISG-PDL compliant. These could be PDL Platform Atomic Services or Platform Composite Services.

4.4.5.3 Optional PDL Platform Services

Services that a PDL platform does not need to include to be considered ETSI-ISG-PDL compliant. Such services should be included if required by the applications running on such platform. These could be PDL Platform Atomic Services or PDL Platform Composite Services.

4.4.5.4 PDL Platform Atomic Services

Services and functionality provided by the PDL platform that are independent of any other Platform Service and do not contain other Atomic or Composite Platform Services. Such services may use external resources that are not a PDL Platform Service offered by said PDL platform (for example, a Location service may use a GPS receiver, an Identity service may use a Certification Authority).

4.4.5.5 PDL Platform Composite Services

Services and functionality provided by the PDL platform that are made up of other Atomic and/or Composite services or other PDL Platform Services offered by said PDL platform (for example, a Security service includes an Identity service).

NOTE: Composite services can be broken down to the Platform services they are composed of. Atomic services cannot be broken down to other Platform services. External services (for example Certification Authority, external storage) are neither Atomic nor Composite as they are external to the platform.

There are several additional categories of Optional Platform Services.

4.4.5.6 Application Specific Platform Services

Services used by specific applications that are not needed or cannot be made useful for other applications (for example, measurement of precipitation is useful for agriculture and weather applications but has no use for data storage applications). Such services will typically be integrated into the specific application that requires them, but the developer and Governance may reach an agreement to include such service as part of the PDL platform to make it useful for other applications or in order to make use of the distributed nature of the PDL.

4.4.5.7 External services

Services provided by an external entity or object and are not part of the Platform, though they may be used by the Platform and by both Atomic and Composite Platform services. As such they are neither Atomic nor Composite.

4.4.6 PDL Abstraction

Consists of a Data Model Broker/Gateway enabling Platform services to communicate with PDL types regardless of the specific type of the underlying PDL. An additional functionality of such abstraction layer is to allow interoperability between different PDL types, which may differ not only in data model structure but also on consensus mechanism and smart-contract functionality. Such abstraction layer hides the differences between PDL types and provides a unified service-facing interface on the services side and a PDL specific interface on the PDL side. This layer is located between the "Techno" and the "Disco" IRPs and implemented through the Data-Model Broker Platform Service where applicable. This is an optional component of the architecture which can be omitted in clauses 4.5.1.2 and 4.5.1.4 platform types as defined in clause 4.5.1.

4.4.7 Endorsed PDL Types

An implementation of one or more PDLs endorsed by ETSI-ISG-PDL.

4.4.8 Interface Reference Points (IRPs)

An IRP is a logical point of interaction which define communication channels through which the Functional Blocks defined above communicate with each other. The IRPs are given names for reference purposes (for example Debka, Tango, etc.).

The IRPs implemented in the ETSI-ISG-PDL RA and their functionality are listed in Table 2.

Table 2: List of IRPs used in the PDL Reference Architecture

IRP	Connects	to	Used for
Debka	Entity A	Entity B	Exchange of operational and commercial data between entities that is not handled using the PDL platform. This functionality is identical to the "Sonata" IRP as defined in MEF-55.1 [i.1].
Tango	An entity	One or more applications	Exchange of application specific information between the application and the entity's existing platforms (e.g. BSS/OSS).
Rumba	An application	Application Abstraction Layer	Exchange of application specific information between the application and the Application Abstraction Layer.
Samba	The Application Abstraction Layer	The platform services layer	Exchange of non-specific information between the Application Abstraction Layer and the Platform Services Layer. See note 1.
Minuet	An application	External objects, services and entities that are not part of the PDL platform	Exchange of information between applications and objects/services/entities that are not part of the PDL Platform.
Bourree	A platform Service	A platform Service	Exchange of information between Platform services.
Hora	Platform Services Layer	External objects, services and entities that are not part of the PDL platform	Exchange of information between the Platform Services layer and objects/services/entities that are not part of the PDL Platform. See note 2.
Techno	Platform Services Layer	PDL Abstraction Layer	Exchange information between the Platform Services Layer and the PDL Abstraction layer.
Disco	PDL Abstraction Layer	PDL Layer	Exchange information between the PDL Abstraction Layer and the PDL Layer. See note 3.
Rondo	PDL Layer	External objects, services and entities that are not part of the PDL platform	Exchange of information between the PDL layer and objects/services/entities that are not part of the PDL Platform.
Waltz	PDL Chain (A)	PDL Chain (B)	Exchange information across different PDL chains implemented in the PDL Platform.
NOTE 1: In platforms where the (optional) Application Abstraction Layer is not implemented, the Rumba and Samba IRPs become one and exchange information between applications and the platform services layer. In such case the name Samba shall be used. NOTE 2: The information may be routed via the Data Model Broker service to overcome data model discrepancies between the Platform Service and external object. NOTE 3: In platforms where the (optional) PDL abstraction layer is not implemented, the Techno and Disco IRPs become one and exchange information between the platform services layer and the PDL layer. In such case the name Disco shall be used.			

4.4.9 Internal and External IRPs

4.4.9.1 Definition

An IRP defines a *message channel*, which is a dedicated communications path connecting two endpoints that has specific associated semantics.

ETSI-ISG-PDL defines two types of IRPs, see below.

4.4.9.2 External IRPs

IRPs that are used for communications between external objects that are not part of the platform and internal objects. The following IRPs are External: Rumba, Tango, Debka, Minuet, Hora, and Rondo.

4.4.9.3 Internal IRPs

IRPs that are used for communications between objects internal to the platform. The following IRPs are Internal: Samba, Techno, Bouree, Waltz.

4.4.9.4 IRP related notes

NOTE 1: The "Disco" IRP may be considered an Internal or an External IRP depending on the implementation. When the Application and PDL are implemented on the same node (physical or logical/virtual) it will be an Internal IRP. When it is implemented on different nodes it becomes an External IRP. As of today, the BWR implementation (as well as the majority of PDL implementations) are following the "Alpha" or "Charlie" architecture and as such both the PDL layer and the platform services layer (and often also the application layer) are implemented on the same node thus the Disco IRP in such implementations is internal.

NOTE 2: The "Debka" IRP is equivalent to the "Sonata" IRP on the MEF-55 LSO Reference Architecture [i.1].

NOTE 3: IRPs may be used to convey data, management, and control. As such there may be parallel implementations of each IRP, one for each function, or a single implementation for all functions. The current version of the present document does not prescribe one way or the other.

4.4.10 Hardware and Software Interfaces

4.4.10.1 Definition

An **Interface** describes the public characteristics and behaviour that specify a software contract for performing a service specific action that is implemented through an IRP. There may be multiple Interfaces implemented on an IRP. ETSI-ISG-PDL will define PDL Software Interfaces, APIs (Application Programming Interfaces), Webhooks and Microservice interfaces, and optionally, hardware interfaces.

There are two types of ETSI-ISG-PDL interfaces, see the following clauses.

4.4.10.2 PDL Software Interface

Defines a point through which communication with a set of resources (for example memory or CPU) of a set of objects is performed. This decouples the implementation of a software function from the rest of the system. It consists of tools, object methods, and other elements of a model and/or code. A commonly used Software Interface is an Application Programming Interface (API) which is a set of communication mechanisms through which a developer constructs a computer program. APIs simplify producing programs, since they abstract the underlying implementation and only expose the objects, and the characteristics and behaviour of those objects that are needed. Other software interfaces may include protocols, DSL (Domain Specific Language), Microservices, and more.

4.4.10.3 PDL Hardware Interface

A point across which electrical, mechanical, and/or optical signals are conveyed from a sender to one or more receivers using one or more protocols. A Hardware Interface decouples the hardware implementation from other Functional Blocks in a system. Examples may include a sensor (for example an IoT device such as a thermometer) connected by wire to a node, an Ethernet cable connected to a node, a fibre-channel connection between a node and directly-attached storage.

4.4.11 IRPs and the Data Model Broker/Gateway

Clause 4.6.3.23 allows different clients (applications, external systems/entities) that use proprietary data models to interact and communicate with the PDL platform using APIs or other communication methods. The Data-Model Broker/Gateway service together with the respective external IRPs ("Tango", "Debka", "Samba", "Rondo", "Hora" and "Minuet") are used to allow such communications.

4.4.12 Architecture-related requirements for a PDL platform.

[R1] A PDL platform **SHALL** include all Mandatory Services.

[R2] A PDL platform **SHALL** include all Optional Services required by applications using such platform.

[O1] A PDL platform **MAY** include Application Specific Services.

[R3] The PDL platform **SHALL** use External Reference Points to communicate to external systems.

NOTE: Platform Services can be either PDL-specific (availability of certain/all features' mandates use of a specific PDL type) or PDL-independent (all features are available on all PDLs compliant with the ETSI-ISG-PDL Reference Architecture).

[D1] PDL Platform Services **SHOULD** be PDL-Independent.

[O2] PDL Platform Services **MAY** be PDL-Specific.

4.5 Development Guiding Principles

4.5.1 Platform development guiding principles

4.5.1.1 Platform Categories

PDL platforms fall into four major categories as defined herewith. Some of those major categories can then be broken down to sub-categories.

a) **Definition:** Platforms that are designed, developed, delivered, and integrated to all users of said platform by a single vendor using a single PDL technology. Such platforms will be labelled as ***"Category Alpha Platforms"*** for the remainder of the present document.

b) **Definition:** Platforms that are designed, developed, delivered, and integrated to all users of said platform by a single vendor, but can operate using two or more underlying PDL technologies. Such platforms will be labelled ***"Category Bravo Platforms"*** for the remainder of the present document.

c) **Definition:** Platforms using a single PDL technology that are designed and developed towards a specification of an Application Abstraction Layer so that any Application that supports such an abstraction layer can interface with said platform in a multi-party and multi-vendor PDL specific environment. Such platforms are labelled as ***"Category Delta Platforms"*** for the remainder of the present document.

d) **Definition:** Platforms that can operate using two or more underlying PDL technologies and are designed and developed towards a specification of an Application abstraction layer so that any Application that supports such an abstraction layer can interface with said platform in a multi-party and multi-vendor PDL agnostic environment. Such platforms are labelled as ***"Category Charlie Platforms"*** for the remainder of the present document.

4.5.1.2 Category Alpha Platform

4.5.1.2.1 Introduction

A Category "Alpha" platform is designed, developed, delivered, and integrated to all users of said platform by a single vendor using a single PDL chain.

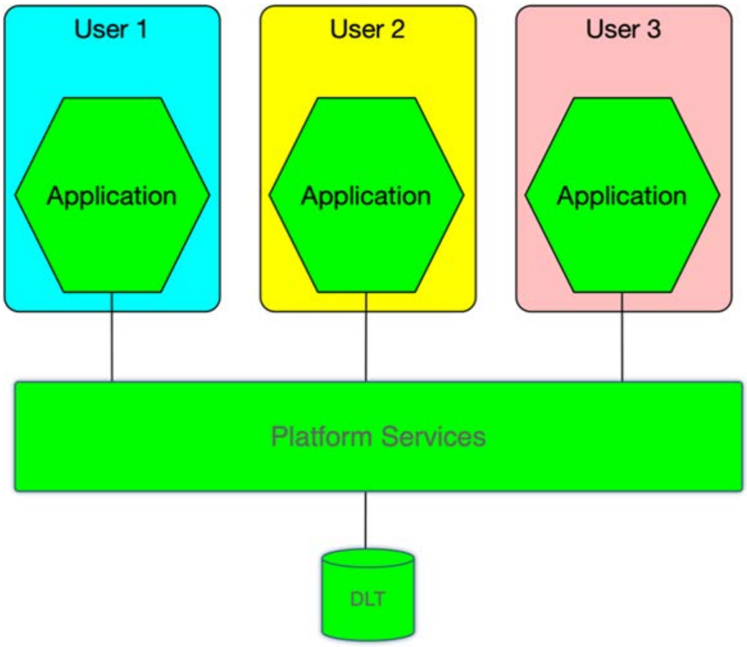

Figure 2: Category Alpha platform

The "Alpha" category is broken down into two options.

4.5.1.2.2 Category Alpha-1 Platform

The PDL and some or all the Platform Services are proprietary to the vendor.

[O3] In a Category Alpha-1 Platform the Platform Services **MAY** include both proprietary and open-source elements.

4.5.1.2.3 Category Alpha-2 Platform

The PDL and all the Platform Services are open-sourced.

[R4] In a Category Alpha-2 Platform all Platform Services **SHALL** be open-sourced.

4.5.1.3 Category Bravo Platform

4.5.1.3.1 Introduction

A Category "Bravo" platform is designed, developed, delivered, and integrated to all users of said platform by a single vendor, but can operate using two or more underlying PDL technologies.

A Category "Bravo" platform includes an abstraction layer between the PDL layer and the Platform Services layer that offers a unified northbound interface between the abstraction layer and the services layer, and a unique, per PDL type, interface between the abstraction layer and the specific PDL types. This abstraction layer is labelled as the ***"PDL Abstraction Layer"*** for the remainder of the present document.

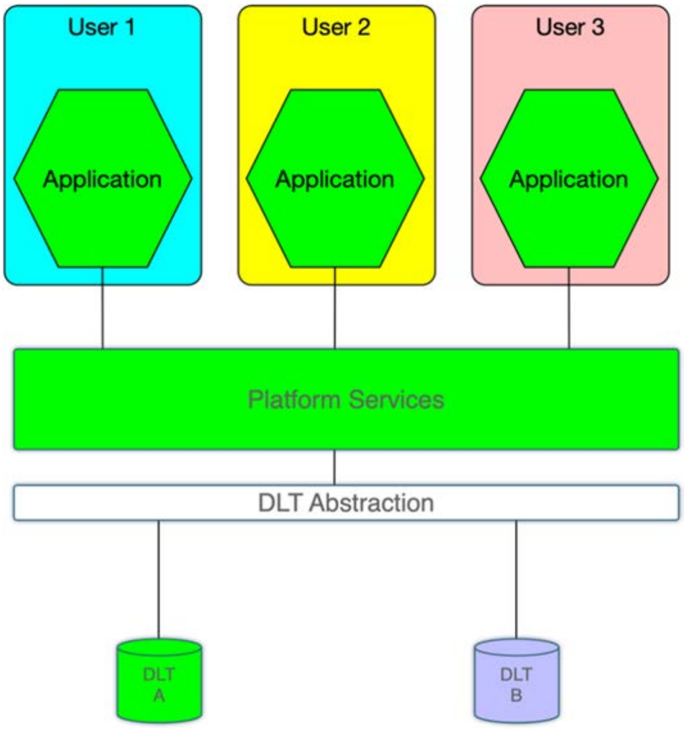

Figure 3: Category Bravo platform

The "Bravo" category is broken down into two options.

4.5.1.3.2 Category Bravo-1 Platform

One or more of the underlying PDL types and some or all the Platform services are proprietary to a vendor.

[R5] In a Category Bravo-1 Platform at least one of the Platform Services **SHALL** be proprietary.

4.5.1.3.3 Category Bravo-2 Platform

The PDLs and all the Platform services are open-sourced.

[R6] In a Category Bravo-2 Platform all Platform Services **SHALL** be open-sourced.

4.5.1.4 Category Charlie Platform

4.5.1.4.1 Introduction

A Category "Charlie" platform is designed and developed towards a specification of an Application Abstraction Layer so that any Application that supports such an abstraction layer can interface with said platform.

This abstraction layer is labelled as the ***"Application Abstraction Layer"*** for the remainder of the present document. The Application Abstraction Layer implements a unified northbound interface between the abstraction layer and the applications using the platform, and a per-platform-specific-service interface between the abstraction layer and the underlying services implemented in the Platform Services layer.

[R7] A Category Charlie platform **SHALL** use a single chain.

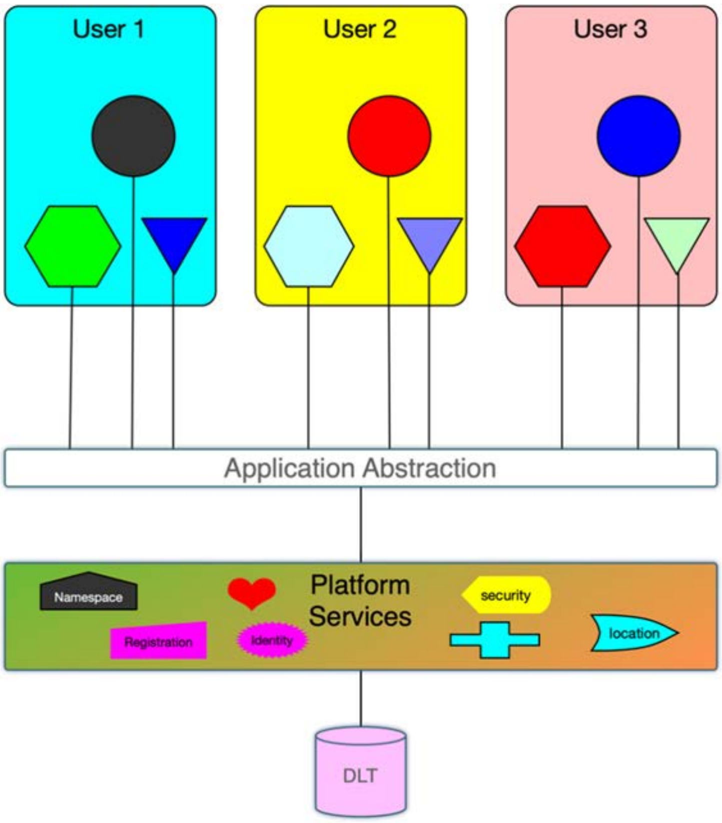

Figure 4: Category Charlie platform

The "Charlie" category is broken down into four options.

4.5.1.4.2 Category Charlie-1 Platform

The platform is being developed and integrated by a single vendor who may integrate third party elements into the platform and may include proprietary elements in the platform.

[O4] In a Category Charlie-1 Platform one or more Platform Services **MAY** be proprietary.

4.5.1.4.3 Category Charlie-2 Platform

The platform is being developed and integrated by a single vendor who may integrate third party elements into the platform and all elements, including the third-party elements, are open-sourced.

[R8] In a Category Charlie-2 Platform all Platform Services **SHALL** be open-sourced.

4.5.1.4.4 Category Charlie-3 Platform

The platform consists of a collection of interoperable modules, each offering one or more of the platform services. Such modules may be developed by different vendors towards service specifications defined or endorsed by ETSI-ISG-PDL. Integration of such modules into an operational platform may be performed by any entity as long as the resulting platform complies with certification tests performed by ETSI-ISG-PDL or a certification entity endorsed by ETSI-ISG-PDL. Some or all the modules may be proprietary.

[R9] In a Category Charlie-3 Platform all Platform Services **SHALL** be compliant with certification tests performed by ETSI-ISG-PDL or a certification entity endorsed by ETSI-ISG-PDL.

[O5] In a Category Charlie-3 Platform one or more Platform Services **MAY** be proprietary.

4.5.1.4.5 Category Charlie-4 Platform

Like Category Charlie-3 Platform but all modules are open-sourced.

4.5.1.5 Category "Delta" Platform

4.5.1.5.1 Introduction

A Category "Delta" platform is designed and developed towards a specification of an Application Abstraction Layer so that any Application that supports such an abstraction layer can interface with said platform, and a PDL Abstraction Layer so that the platform services can operate on any PDL chain endorsed by ETSI-ISG-PDL.

These abstraction layers are labelled as the ***"Application Abstraction Layer"*** and the ***"PDL Abstraction Layer"*** respectively for the reminder of the present document.

The Application Abstraction Layer is using a unified northbound interface towards the Applications implemented using the *Rumba* IRP, and a southbound unified interface towards the Platform Services layer using the *Samba* IRP.

The PDL Abstraction Layer is using a unified northbound interface towards the Platform services layer using the *Techno* IRP and a unified southbound interface towards the PDL layer using the *Disco* IRP.

[O6] The Category Delta platform **MAY** use two or more chains of different types.

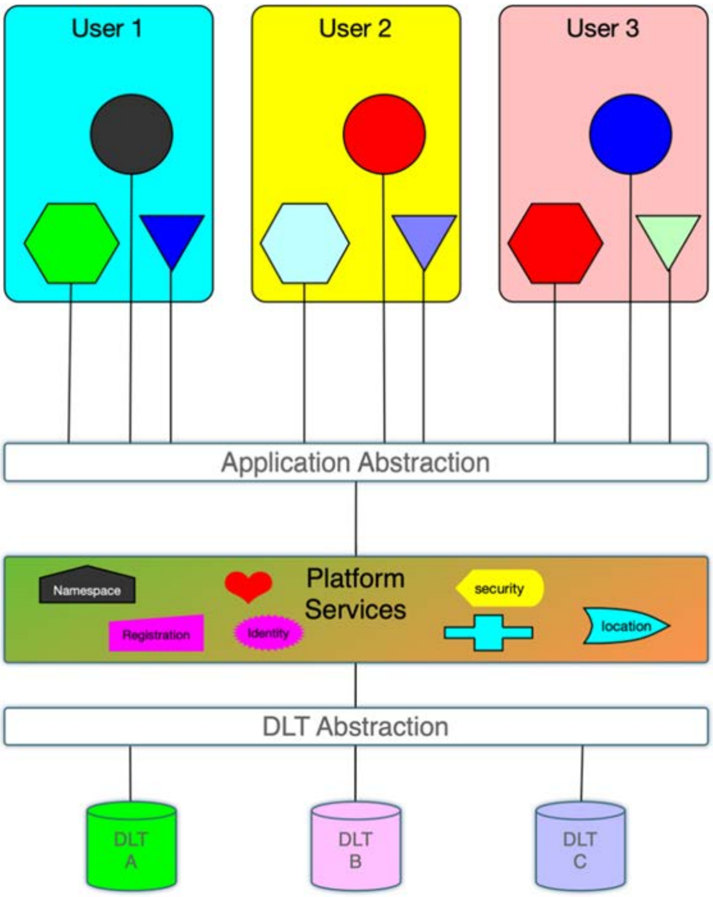

Figure 5: Category Delta platform

The "Delta" category is broken down into four options.

4.5.1.5.2 Category Delta-1 Platform

The platform is being developed and integrated by a single vendor who may integrate third party elements into the platform and may include proprietary elements in the platform.

4.5.1.5.3 Category Delta-2 Platform

The platform is being developed and integrated by a single vendor who may integrate third party elements into the platform and all elements, including the third-party elements, are open-sourced.

4.5.1.5.4 Category Delta-3 Platform

The platform consists of a collection of interoperable modules, each offering one or more of the platform services. Such modules may be developed by different vendors towards service specifications defined or endorsed by ETSI-ISG-PDL. Integration of such modules into an operational platform may be performed by any entity as long as the resulting platform complies with certification tests performed by ETSI-ISG-PDL or a certification entity endorsed by ETSI-ISG-PDL. Some or all the modules may be proprietary.

4.5.1.5.5 Category Delta-4 Platform

Like Category Delta-3 Platform but all modules are open-sourced.

4.5.2 Application development guiding principles

The guiding principles of Application development follow similar logic and categorization of platform development principles:

a) Applications that are developed and delivered to all users of said application by a single vendor using a Category Alpha Platform developed by that same vendor and thus can only use a prescribed PDL type. Such applications will be labelled as ***"Category Alpha Applications"*** for the remainder of the present document.

b) Applications that are developed and delivered to all users of said application by a single vendor using a Category Bravo Platform developed by that same vendor. Such applications will be labelled as ***"Category Bravo Applications"*** for the remainder of the present document. Category Bravo Applications are not limited to a prescribed PDL type and can be implemented using any PDL type supported by the Category Bravo Platform.

c) Applications that are developed towards a specification of an Application so that any user of an application supporting such specifications can fully interoperate with other users of other applications built towards the same Application specifications. Such applications are labelled as ***"Category Charlie Applications"*** for the remainder of the present document. The same logic and nomenclature also apply for ***"Category Delta Applications"***.

NOTE: *Category Delta Applications* are redundant to *Category Charlie Applications* as the Platform Services Layer hides the underlying PDL type.

4.5.3 Platform Services Dependency

Due to the dependency of Composite Platform services on other Platform Services, when a Composite Platform Service is implemented in a certain PDL Platform, and that Composite Platform Service is using an Optional Platform Service, that Optional Platform Service SHALL be implemented on that specific PDL Platform.

[R10] When a Composite Platform Service that is implemented in a PDL Platform is dependent on or made up of an Optional Platform Service, said Optional Platform Service **SHALL** be implemented in that PDL Platform.

4.5.4 Platform Services Plurality

In Category Charlie and Category Delta platforms multiple versions of the same Platform Service can be provided by different vendors. Each application developer and user of the platform may choose the respective version of Platform service that meets their specific requirements and set of features.

[R11] An application developer **SHALL** develop the application, so it is fully compliant with the corresponding set of ETSI-ISG-PDL specifications and requirements for that service.

[O7] An application developer **MAY** add feature that exceed the ETSI-ISG-PDL requirements.

[D2] When development of additional features for a service requires collaboration between multiple entities, the developer **SHOULD** propose such enhancements through designated ETSI-ISG-PDL groups or committees.

4.5.5 Abstraction Layer Implementation

An Abstraction Layer is an abstract structure that serves as an intermediator between subsystems that may be using different vocabulary and methods that serves their respective purposes. As discussed earlier, a platform may include up to two abstraction layers: An *Application Abstraction Layer* and a *PDL Abstraction Layer*. The functionality of an abstraction layer is implemented by routing all ingress and egress communications traversing through the external IRPs to the Data-Model Broker/Gateway Platform Service. A typical implementation is described in Figure 7 herewith.

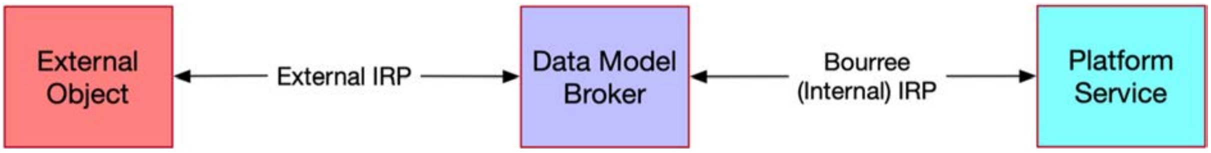

Figure 6: Abstraction Layer Implementation

The "External Objects" depicted above may be any object external to the Platform Services Layer (for example an application, a PDL, external storage, an external service/platform).

4.6 Platform Services

4.6.1 List of all Platform Services

As discussed in clause 4.4, the Platform Services is a set of modular Functional Blocks that are either PDL Platform Services themselves (Atomic Services) or are used to create PDL Platform Composite services. To maximize reusability, Composite services are built using composition and appropriate software patterns [i.3]. This enables improved components of a composition to be used without affecting other services.

Table 3 below lists all Platform services.

Table 3: List of Platform Services

PDL Platform Service name	Mandatory (M) or Optional (O)	Atomic (A) or Composite (C)	Location in the present document (clause number)	Short description
Namespace	M	A	4.6.2.2	Ensures that all of a given set of objects for a particular function have unique names.
Identity	M	A	4.6.2.3	Unambiguously identifies an instance of an entity from all other instances of this and other objects.
Location	O	A	4.6.2.4	Associates an object with a location.
Registration	O	A	4.6.2.5	List a managed object with authorities or registries.
Discovery	O	A	4.6.2.6	Discovery of services offered by the services layer and discovery of PDL networks.
Messaging	M	C	4.6.3.2	Enables communication between a group of entities.
Policy	O	C	4.6.3.3	Manage and control the changing and/or maintaining of the state of managed objects.
Security	M	C	4.6.3.4	A collection of services that assess, reduce, protect, and manage security risks.
Authentication	M	C	4.6.3.4.2	Verifies that a subject requesting to perform an operation on a target is who they say they are.
Authorization	O	C	4.6.3.4.3	Permitting or denying access to a target by a subject.

PDL Platform Service name	Mandatory (M) or Optional (O)	Atomic (A) or Composite (C)	Location in the present document (clause number)	Short description
Cryptography	O	C	4.6.3.4.4	Managing protocols that prevent third parties from reading private communications.
Encryption	O	C	4.6.3.4.5	Encoding information using a key into an unintelligible form.
Identity Management	O	C	4.6.3.4.6	Access control based on the identity of an entity or object.
Key Management	O	C	4.6.3.4.7	Management of cryptographic keys.
Logging	O	C	4.6.3.5	Dynamic ingestion and collection of logs.
Governance	M	C	4.6.3.6	Rules and tools that control the behaviour and function of a PDL.
Implementation Agreements	O	C	4.6.3.6.2	Rules and agreements that describe how Services are implemented and control the behaviour of a PDL platform.
Governing Entity	M	C	4.6.3.6.3	Defines the rules and implementation agreements. Ensures compliance. Resolves conflicts where needed.
Composition	O	C	4.6.3.7	Defines who can compose new services and how such new services are composed.
Access Control	M	C	4.6.3.8	Defines who can perform which operations on which set of target entities.
Fault Tolerance	O	C	4.6.3.9	Defines how to handle faulty instructions.
Distribution Transparency	O	C	4.6.3.10	Defines how to maintain transparency when distributing information to target entities.
Publish and Subscribe	O	C	4.6.3.11	Defines how entities publish services and subscribe to services.
Concurrency	O	C	4.6.3.12	Defines how entities handle concurrency.
Storage	M	C	4.6.3.13	A group of services related to Storage.
In Memory Storage	M	C	4.6.3.13.2	Data that is stored in the RAM of a computer running an application.
File System Storage	M	C	4.6.3.13.3	Storage on a directly connected storage device.
On-Chain Storage	M	C	4.6.3.13.4	Application data that is stored in blocks on all nodes using the chain.
Off-Chain storge	O	C	4.6.3.13.5	Information in a digital, machine-readable medium that is not stored on the main chain.
Distributed Blockchain Storage	M	C	4.6.3.13.6	Storage on a Distributed Blockchain ledger.
Modelling	M	C	4.6.3.14	A group of services related to Modelling.
Information Model	M	C	4.6.3.14.2	Presentation of concepts of interest to platform management environment in a technology-neutral form as objects and relationships between objects.
Data Model	M	C	4.6.3.14.3	Representation of applicable concepts in a technology-specific concrete form.
Model Search	O	C	4.6.3.14.4	Enables search for specific or generic models within existing information and data models.
Model Stitching	O	C	4.6.3.14.5	Enables integrating multiple models or parts of models into a single model.
Topology	M	C	4.6.3.15	Allows a node to identify other nodes on the PDL and identify which nodes to communicate with when performing PDL related tasks.
Event Processing	M	C	4.6.3.16	Processes node-specific and platform-wide events as they occur.

PDL Platform Service name	Mandatory (M) or Optional (O)	Atomic (A) or Composite (C)	Location in the present document (clause number)	Short description
Distributed Data Collection	O	C	4.6.3.17	Performs tasks related to collection of data that are location-independent.
Distributed Secret Sharing	O	C	4.6.3.18	Sharing of confidential data between nodes in a manner that maintains confidentiality of the data.
Resource Management	M	C	4.6.3.19	Defines how to administer and manage Resources.
Resource Discovery	O	C	4.6.3.19.2	Enables discovery of resources available to applications and nodes.
Resource Virtualization	O	C	4.6.3.19.3	Creating a virtual resource that mimics the behaviour of a physical resource.
Resource Inventory Management	O	C	4.6.3.19.4	Management of node-specific and platform-wide resource inventory.
Resource Admin and Management	M	C	4.6.3.19.5	Administration and management of node-specific and platform-wide resources.
Resource FCAPS	O	C	4.6.3.19.6	Resource management tasks defined by the ISO model.
Resource Composition	O	C	4.6.3.19.7	Creation and management of composite resources.
Platform Services Management	M	C	4.6.3.20	Defines how to administer and manage Platform Services.
Platform Service Discovery	M	C	4.6.3.20.2	Provides means to discover services available to applications and nodes.
Platform Service Virtualization	O	C	4.6.3.20.3	Creating a service using virtual resources.
Platform Service Inventory Management	O	C	4.6.3.20.4	Keeping track of inventory and serviceability of Platform services.
Platform Service Admin and Management	M	C	4.6.3.20.5	Administration and management of Platform Services through governance.
Platform Service FCAPS	O	C	4.6.3.20.6	Platform Service management tasks defined by the ISO model.
Platform Service Composition	O	C	4.6.3.20.7	Creation and management of the composition of Composite Platform Services.
Application Management	M	C	4.6.3.21	Creation and management of Applications.
Application Composition	M	C	4.6.3.21.2	Composing an application from two or more managed objects.
Application and Service Orchestration	O	C	4.6.3.21.3	Orchestrating multiple managed objects so they provide a desired set of behaviours.
Orchestration	O	C	4.6.3.21.4	Orchestration of objects, resources, services, and/or applications so that they collectively provide the desired functionality and behaviour.
Platform Exploration	O123	C	4.6.3.21.5	Allows an application to indicate its requirements and explore whether the platform offers such service capabilities
Application Registration	O	C	4.6.3.21.6	Registers and lists all applications operated on a platform.
Transaction Management	O	C	4.6.3.22	Facilitates transaction related interactions between applications/services and underlying PDL or PDLs.
Data Model Gateway/Broker	O	C	4.6.3.23	Defines tools that enable two systems with different data models to interact.

PDL Platform Service name	Mandatory (M) or Optional (O)	Atomic (A) or Composite (C)	Location in the present document (clause number)	Short description
API Presentation	O	C	4.6.3.23.2	A specific Data Model Gateway/Broker implementation for environments that use APIs to exchange data between objects (including micro-services).
Application Specific Services	O	C	4.6.3.24	Serve a specific application or a group of applications but not required or used by other applications using the platform.
Accounting Service	O	C	4.6.3.25	Measure consumption of resources and services by users and generate a usage report.

4.6.2 Atomic Platform Services

4.6.2.1 Introduction to Atomic Platform Services

The Atomic Services are a set of PDL Platform Services that other Platform Services may use, either directly or indirectly. Atomic Platform Services do not use any other PDL Platform Service but may use services external to the PDL platform.

[R12] Atomic Platform Services **SHALL NOT** use any other Platform Service to fulfil their functionality.

[O8] Atomic Platform Services **MAY** use services external to the PDL Platform.

There are five (5) PDL Atomic Platform Services, four (4) of which are also Mandatory Platform Services. They are shown in Figure 7 herewith.

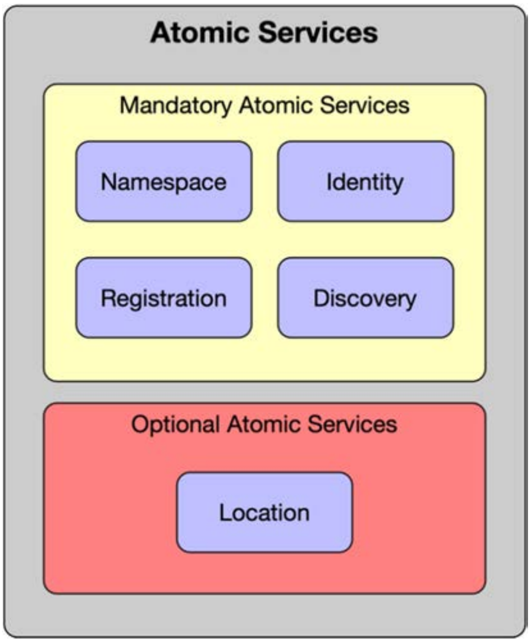

Figure 7: Atomic Platform Services

4.6.2.2 Namespace Platform Service

The Namespace Platform Service ensures that all of a given set of objects for a particular function have unique names so that they can be easily identified. This enables multiple internal and external domains to communicate and interact with each other while avoiding name collisions between multiple identifiers that share the same name for a given object. Examples of internal domains are different administrative domains within an organization (for example engineering and sales), while examples of external domains include different partners (for example service and content providers) of an organization.

[R13] A Namespace Platform Service **SHALL** provide a unique name for each managed object in its models that distinguishes each object instance from all other object instances (including multiple instances of the same object) that it contains.

Namespaces provide a scope for object names. Namespaces are typically structured as hierarchies to allow reuse of names in different contexts. Examples include file systems and DNS. A namespace is a *scoping container*. Examples include application container and messaging container services.

[D3] A Namespace Platform Service **SHOULD** support hierarchical names.

Namespaces may be simplified by using consistent prefixes for each namespace.

[D4] A name in a Namespace **SHOULD** consist of a namespace identifier and a local (to that namespace) unique name.

4.6.2.3 Identity Platform Service

The Identity of an entity is a set of context-dependent digital identifiers that unambiguously identify an instance of that entity from all other instances of this and other objects. An identity may require multiple attributes to uniquely identify it (for example two products with the same name have other different attributes, such as different serial numbers).

[R14] An identity **SHALL** be constructed using one or more context-dependent digital identifiers that enable an object instance to be unambiguously identified.

A digital identifier is a secure object that is unique within a particular namespace. It is recommended that every digital identifier is assigned a namespace.

[D5] A digital identifier **SHOULD** be defined within a namespace to guarantee its uniqueness.

An entity may be used in different situations. Therefore, the same entity may be identified using a different set of digital identifiers for each situation. This enables the semantics of the use of an entity in each situation to be considered.

[O9] A Managed Object **MAY** have multiple context-dependent digital identifiers for establishing the Identity of that Managed Object in different situations in which it is used.

AN ETSI-ISG-PDL Identity Service provides a single identity token per instance of an entity for all services so that this instance is identified unambiguously and in the same manner by all services.

[R15] An Identity Service **SHALL** provide a single digital identity token per instance of an entity.

4.6.2.4 Location Platform Service

The location of an entity may or may not be relevant to the function of a PDL or a service, thus this Atomic Platform Service is optional. In applications and scenarios where location is of essence, it may affect factors such as network latency (and the resulting transaction speeds), governing laws and regulations, costs, access restrictions and more. There are multiple methods of defining locations. There are physical addresses (for example GPS longitude/latitude coordinates, street addresses, postal codes, building names), relative addresses (for example 50 meters east of the main gate, and Virtual locations (for example IP address, Telephone number, MAC address). Certain location descriptors are more accurate than others (for example a postal code may relate to a whole street while GPS coordinates may define a location with an accuracy of a few meters).

[O10] A Managed Object **MAY** be associated with a set of locations.

[R16] The location of a Managed Object associated with a location **SHALL** be represented in a method understood in the respective geography where it is located.

[R17] The location of a Managed Object associated with a location **SHALL** be defined using a location method compliant with the level of accuracy required by the respective application.

4.6.2.5 Registration Platform Service

Registration services provide means to list a Managed Object with local or international authorities or registries. Such registries allow reference to such Managed Objects for legal, commercial, and operational purposes. Registration requirements vary with geography, though not all registries are linked to the geography in which they are used. Certain Managed Objects (for example a PDL serving a geographically diverse application) operate in multiple geographies and may require multiple registrations.

[O11] A Managed Object **MAY** be registered in one or more registries.

[R18] A registered Managed Object **SHALL** be registered in accordance with the regulations and rules applicable in the geographies in which it operates.

4.6.2.6 Discovery Platform Service

Discovery services provide means to:

a) Discover Platform Services offered by a Platform Services layer; and/or

b) Discover a registered PDL entity.

For example, an application can discover the Platform Services available on a PDL Platform. In another example, Platform Service can discover an underlying PDL network, which has been registered to a Platform Services layer.

[R19] A Platform Services Layer **SHALL** have a Discovery Service.

[R20] A Discovery Service **SHALL** support discovery of Platform Services on the Services Layer.

[R21] A Discovery Service **SHALL** support discovery of PDL networks that have been registered to the DLT Layer.

[R22] A Discovery Service **SHALL** support discovery of PDL Managed Objects that have been registered to the Services Layer.

4.6.3 Composite Services

4.6.3.1 List of all Composite platform Services

The Composite Platform Services are a set of Functional Blocks that provide services that other Platform Services use, either directly or indirectly. They use one or more other Platform Services to fulfil their functionality. Composition allows building more complex architectural concepts and functions. There is a total of 53 Composite Platform Services, 16 of which are Mandatory. Services are grouped into sub-groups by their function for reference purposes but are non-hierarchical. Any Platform Service or application may use any other Platform service. They are shown in Figure 8 below.

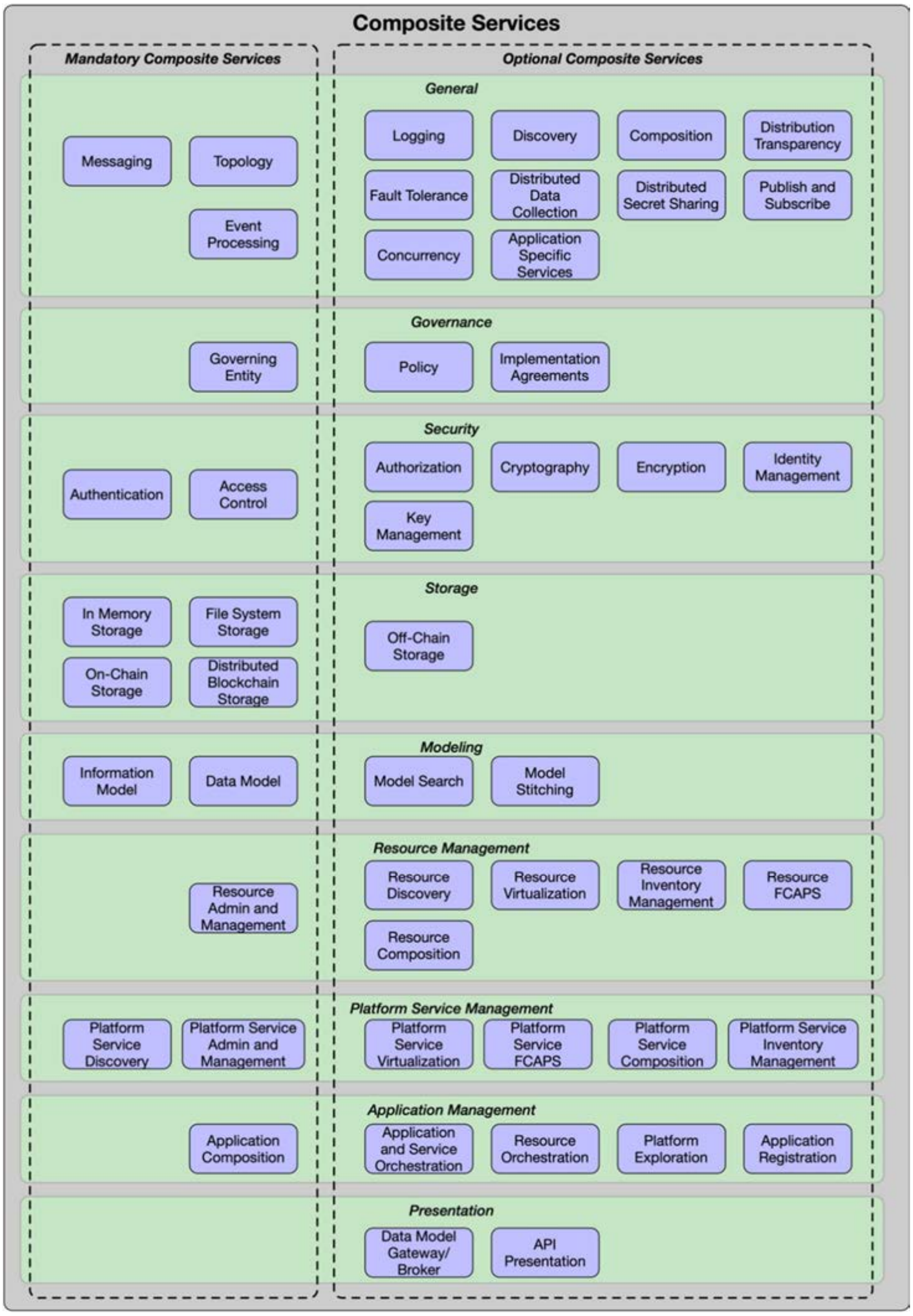

Figure 8: Composite Platform Services

4.6.3.2 Messaging Service

A Messaging Service enables communication between a group of entities (for example PDL nodes, Application users, Platform Services). A message is a discrete unit of communication, sent by a *producer* and received by a *consumer*. There are two fundamentally different types of messaging:

a) Synchronous communication, which is a *tightly coupled* solution to exchange information (for example opening a socket over a connection-oriented protocol such as TCP/IP and transmitting data through it). In Synchronous communications messages are delivered in real time and are immediately read by the recipient.

b) Asynchronous communication, which is a *loosely coupled* solution that minimizes producer and consumer dependencies. In asynchronous communications there may be a gap between the time a message was sent until it had been read by the recipient, and in certain scenarios the recipient does not even have to read the message.

Synchronous messaging is tightly coupled because of its main three dependencies: *temporal* (all components have to be available at the same time), *location* (each component has to know the address of each other component), and *data structure* (all components have to agree on the data format and on the binary representation). Asynchronous messaging acts as an indirection layer among entities that want to communicate, removing the above three dependencies.

[R23] The Messaging Framework Service **SHALL** support asynchronous communications.

[O12] The Messaging Framework Service **MAY** support synchronous communications.

[CR<O12/1] All components involved in a Messaging Framework Service that supports synchronous communications **SHALL** be available at the same time.

[CR<O12/2] All components involved in a Messaging Framework Service that supports synchronous communications **SHALL** know the address of each other component.

[CR<O12/3] All components involved in a Messaging Framework Service that supports synchronous communications **SHALL** use a uniform data format and a uniform binary representation.

There are two types of asynchronous communication models. A *Message Broker* is a centralized system that receives messages, determines the correct destination for each message, and sends the message to that destination. A *Message Bus* enables interacting entities to communicate using a set of shared interfaces.

[R24] The Messaging Framework Service **SHALL** support a Message Bus.

[O13] The Messaging Framework Service **MAY** support a Message Broker.

4.6.3.3 Policy Service

A Policy is a set of rules that is used to manage and control the changing and/or maintaining of the state of one or more managed objects. A Policy Service is a collection of technologies that enable policies to be created, validated, read, updated, deleted, and managed. ETSI-ISG-PDL clients shall use policies to interact with the platforms.

[R25] ETSI-ISG-PDL compliant client implementations **SHALL** use policies to communicate and interact with the platform.

Policies are used in two important ways. First, they enable a consistent and auditable delivery mechanism for requesting and receiving data, and performing commands, to be implemented. Second, policies provide a common communications mechanism for exchanging information and commands.

[R26] Components of a distributed implementation of the platform **SHALL** use policies to exchange information and commands.

[D6] ETSI-ISG-PDL compliant client implementations **SHOULD** use policies for requesting services of, and exchanging information with, the platform.

[R20] Policies **SHALL** be defined, maintained, and enforced by the Governance.

4.6.3.4 Security Platform Services

4.6.3.4.1 Introduction to Security Platform Services

A Security Service is a collection of security technologies that assess, reduce, protect, and manage security risks. These technologies are atomic in nature. This means that a category such as access management is included, since different types of access management solutions (for example MAC, DAC, ABAC and RBAC) use different technologies, but all serve the same fundamental service. In contrast, solutions such as Zero Trust or SASE are NOT included as A Platform Service because both are constructed from other security related Platform Services.

The basic Security Platform Services are shown in Figure 9.

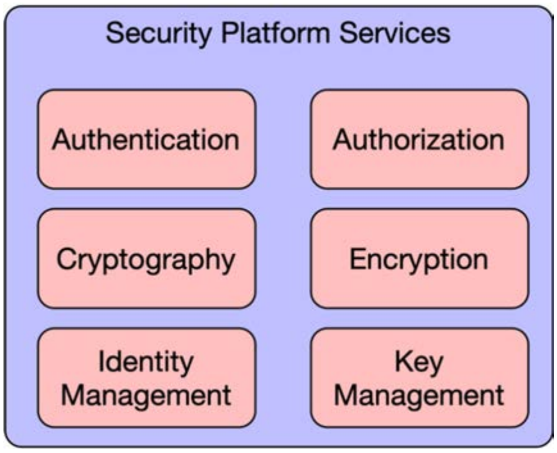

Figure 9: Security Platform Services

4.6.3.4.2 Authentication Platform Service

Authentication is the process of verifying that a subject requesting to perform an operation on a target is who they claim to be. Policies may be used to dictate the set of verification criteria used. The Authentication Platform Service depends on the Namespace Platform Service and the Identity Platform Service.

[R28] An Authentication Platform Service **SHALL** be implemented in all platforms.

[O14] Policies **MAY** be used to dictate the set of verification criteria used for authentication.

4.6.3.4.3 Authorization Platform Service

Authorization is the process that results in permitting or denying access to a target by a subject. Policies may be used to prescribe the criteria for the authorization decision. The Authorization Platform Service depends on the Namespace Platform Service and the Identity Platform Service.

[R29] An Authorization Platform Service **SHALL** be implemented in all platforms.

[O15] Policies **MAY** be used to prescribe the criteria for the authorization decision.

4.6.3.4.4 Cryptography Platform Service

Cryptography is the process of constructing and verifying protocols that prevent third parties from reading private communications. The Cryptography Platform Service depends on the Namespace Platform Service and the Identity Platform Service.

[R30] A Cryptography Platform Service **SHALL** be implemented in all platforms.

4.6.3.4.5 Encryption Platform Service

Encryption is the process of encoding information using a key into an unintelligible form to protect sensitive information. The unintelligible form of the information SHALL be decrypted using the key to recover the original information. The Authentication Platform Service depends on the Namespace Platform Service and the Identity Platform Service.

[R31] An Encryption Platform Service **SHALL** be implemented in all platforms.

4.6.3.4.6 Identity Management Platform Service

Identity Management defines access control based on the identity of an entity or an object that initiates a particular set of operations on a target according to a set of criteria. The Identity Management Platform Service depends on the Namespace Platform Service and the Identity Platform Service.

[R32] An Identity-Management Platform Service **SHALL** be implemented in all platforms.

NOTE: The *Identity Management Platform Service* and the *Identity Platform Service* are two distinct and different services. The Identity Platform service defines how identities are assigned, while the Identity Management Platform Service defines how access is managed based on an assigned identity.

4.6.3.4.7 Key Management Platform Service

Key management refers to management of cryptographic keys in a cryptosystem. This includes dealing with the generation, exchange, storage, use, destruction, and replacement of keys. It also includes cryptographic algorithm and protocol design. The Authentication Platform Service depends on the Namespace Platform Service and the Identity Platform Service.

[R33] A Key Management Platform Service **SHALL** be implemented in all PDL platform.

4.6.3.5 Logging Platform Service

A Logging Service is a collection of technologies that enable different types of logs to be ingested and collected dynamically. The Logging Service may provide an optional normalization service, which enables related logs generated by different sources using different technologies to be normalized into a single data model.

[R34] A Logging Platform Service **SHALL** be implemented in all platforms.

[O16] When a Logging Platform Service is implemented it **MAY** also provide a normalization service.

4.6.3.6 Governance Platform Services

4.6.3.6.1 Introduction to Governance Platform Services

Governance Platform Services are a collection of rules and tools that control the behaviour and function of a PDL Platform. The implementation and enforcement of the rules is carried out using other Platform Services. The Governance Platform Services are depicted in Figure 10 herewith.

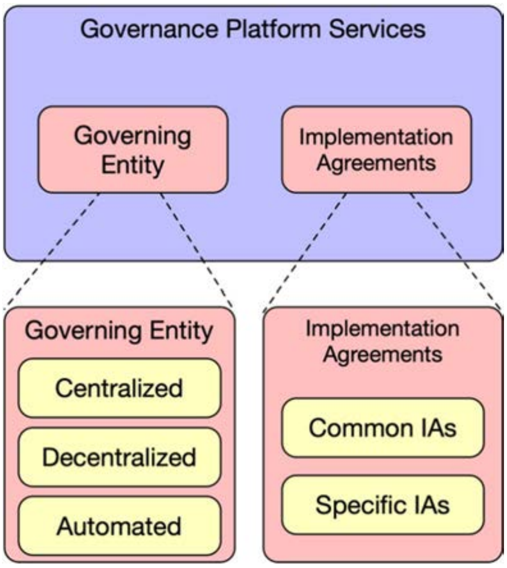

Figure 10: Governance Platform Services

Governance is divided to two functions:

a) **Implementation Agreements (IAs):** A collection of rules and agreements that describe how Services are implemented and control the behaviour of the PDL platform. In a Category Charlie/Delta platform such agreements and rules are typically developed in a collaborative manner by the participants of the PDL platform and are implemented by the platform/service/application developers. They would typically be prescribed by the governing entity and implemented by the developer in a Category Alpha and Category Bravo type platforms.

b) **Governing Entity:** An entity that performs governance tasks by defining the rules and IAs, as well as ensuring compliance and resolving conflicts where needed. Governance also defines the methods by which the Governing Entity is established, its composition and the methods by which it defines/accepts rules/IAs and enforces compliance.

4.6.3.6.2 ETSI-ISG-PDL Implementation Agreements

4.6.3.6.2.1 Definition of Implementation Agreements

Rules and agreements that describe how the Services defined in the present document are implemented. Such rules and agreements define the specific details and methods used to implement the services. For example, the choice of a specific PDL chain type or the acceptance criteria of entities to the PDL platform.

Implementation Agreements are divided to three groups.

4.6.3.6.2.2 Common Implementation Agreements - Common Rules

Common IAs are those that are used by all applications, users and entities involved with a PDL platform and the same rules are applicable to all. For example, a PDL platform may require that all entities and applications use a specific Identity Service and specific security methods.

[R35] All applications, users and entities using a PDL platform **SHALL** comply with all Common Implementation Agreements - Common Rules.

4.6.3.6.2.3 Common Implementation Agreements - Specific Rules

These types of IAs are common to all applications, users and entities involved with a PDL platform, but the specific rules are application- or jurisdiction-dependent. For example, a PDL platform may require that all entities and applications use a specific Location Service, but different geographies may use different methods to define a location and different applications may require different granularity/accuracy location information.

[R36] All applications, users and entities using a PDL platform **SHALL** comply with all Common Implementation Agreements - Specific Rules.

[O17] The specific rules **MAY** vary depending on the specific user, application, and/or jurisdiction.

4.6.3.6.4.4 Specific Implementation Agreements

Those are IAs that are specific to one or more application or jurisdiction and only apply to entities subject to such jurisdiction and/or using such applications. For example, EU/EE entities are subject to GDPR; thus, any application operated by an entity subject to EU/EEA jurisdiction will have to use an Implementation Agreement that complies with GDPR. Another example is a requirement that all applications involved with monetary transactions use a specific encryption method prescribed by the governance.

[D7] All applications, users and entities involved with a PDL platform **SHOULD** comply with all Specific Implementation Agreements.

[O18] Specific Applications and Entities **MAY** be exempt from compliance with Specific Implementation Agreements.

4.6.3.6.3 Governing Entity

4.6.3.6.3.1 Governing Entity types

The Governing Entity performs governance tasks by defining the rules and AIs, as well as ensuring compliance and resolving conflicts where needed. Governance also defines the methods by which the Governing Entity is established, its composition, the methods by which it defines/accepts rules/IAs and enforces compliance and the legally binding agreements that need to be signed by entities and individuals.

There are several types of Governing Entities.

4.6.3.6.3.2 Centralized governance

Centralized Governance is a scenario where the Governing Entity is chosen or agreed upon by the PDL participants.

[R21] The Governing Entity in a Centralized Governance scenario **SHALL** be elected through consensus.

[D8] The Governing Entity in a Centralized Governance scenario **SHOULD** be an entity participating in the PDL Platform.

[O19] The Governing Entity in a Centralized Governance scenario **MAY** be an external entity not participating in the PDL Platform.

[O20] The Governing Entity in a Centralized Governance scenario **MAY** consist of more than a single entity.

[R22] When the Governing Entity in a Centralized Governance scenario consists of more than a single entity all its decisions **SHALL** be reached through consensus between the entities the Governing Entity consists of.

4.6.3.6.3.3 Decentralized governance

Decentralized Governance is a scenario where the Governing Entity consists of all PDL participants or a group of representatives thereof. Governance tasks are performed using PDL consensus and policies.

[R23] The representatives participating in a Governing Entity in a Decentralized Governance scenario **SHALL** be elected through consensus.

[R40] The Governing Entity in a Decentralized Governance scenario **SHALL** use PDL consensus and Implementation Agreements to perform governance tasks.

[D9] The Governing Entities in a Decentralized Governance scenario SHOULD each be entities that are participating in the PDL Platform.

4.6.3.6.3.4 Automated governance

Automated governance is a scenario where decisions, consensus and policy enforcement are taken by software (pre-programmed or Artificial Intelligence) on behalf of the PDL participants.

[R41] Changes to the governing software in an Automated governance scenario **SHALL** be accepted through consensus of the PDL participants.

4.6.3.6.3.5 Other types of governance

Additional governance structures may be formed in the future and documented in a future version of the present document.

4.6.2.6.4 Creating, Changing and Enforcing Governance IAs and rules

Governance IAs and rules are created, maintained, changed, and enforced by the Governing Entity using consensus. It is recommended that as many of the governance tasks as possible be handled automatically, but some tasks may require manual/human intervention.

[D10] Governance tasks **SHOULD** be performed automatically.

[O21] Governance tasks **MAY** be performed manually.

[R42] Any formal or official communication between PDL participants **SHALL** be routed through, monitored, and recorded by the governance.

The above requirement can be fulfilled by storing all such communications on-chain in a manner readable by the governance.

NOTE: The above requirement does not imply that all communications between all entities are routed through and recorded by the governance. It does imply that official and formal information, such as contracts, smart contracts, rate plans and any other information that is required by the governance to fulfil its mission is made available to the governance.

4.6.3.7 Composition Platform Service

Composition Platform Service defines which *subject* entities can compose new objects and how such new objects are composed from other objects.

The requirements related to composition of different object types are listed in the below clauses:

a) 4.6.3.19.7.

b) 4.6.3.20.7.

c) 4.6.3.21.2.

4.6.3.8 Access Control Platform Service

Access Control Service defines which *subject* entities can perform which operations on which set of *target* entities according to a set of criteria.

There are four established Access Control Policies in use today [i.2]:

a) MAC (Mandatory Access Control)

b) DAC (Discretionary Access Control)

c) RBAC (Role Based Access Control)

d) ABAC (Attribute Based Access Control)

It is beyond the scope of the present document to go into discussion of the differences between those policies.

A Policy is a set of rules that is used to manage and control the changing and/or maintaining of the state of one or more managed objects. An Access Control Policy defines the privileges and permissions of a subject entity to perform a set of requested operations on a set of target entities. Policy Based Access Control (PBAC) defines the type of Policies required to implement the appropriate type of access control methodology according to the needs of the PDL Platform.

[R43] Access Control Services **SHALL** use Access Control Policies to manage access control for the entities that it protects.

[O22] Access Control Services **MAY** use Policies to extend the functionality of standardized Access Control approaches.

4.6.3.9 Fault Tolerance Platform Service

Fault Tolerance Service defines how a PDL Platform behaves in two scenarios:

a) ***Faulty Instructions***: How *target* entities handle situations where faulty instructions are being given by *subject* entities according to a set of criteria. Faulty instructions may include, but are not limited to, violation of consensus, violation of security protocol, violation of policy. Faulty instructions may also result from incorrect or inaccurate data ingestion (for example a thermometer giving inaccurate readings, or a communications error due to congestion). It is highly recommended, though not mandated, that a platform establishes processes, automated or manual, to overcome such faults while retaining the platform's integrity.

b) ***Faulty Components***: The ability of a PDL Platform to continue operating correctly when one or more of its components fails.

[D11] Reasonable measures **SHOULD** be taken to allow a platform to continue operations in presence of a level of faults that is below a pre-defined threshold.

[R44] Measures **SHALL** be taken to notify entities using a platform when faults exceed a pre-defined threshold.

4.6.3.10 Distribution Transparency Platform Service

Distribution Transparency Service defines how *subject* entities maintain transparency when distributing information to *target* entities. Transparency is defined by Policy and may vary depending on location, regulation, and application. For example, a commercial agreement between two entities may include confidential commercial information that should not be visible to other parties; yet those other parties may need to be aware that a commercial agreement exists between said entities. Under such scenario the Distribution Transparency service will have to ensure the details of the involved parties are visible to other parties, while the confidential parts of the agreement are encrypted and cannot be read.

[R45] A platform **SHALL** maintain Distribution Transparency in accordance with all applicable Policies.

4.6.3.11 Publish and Subscribe Platform Service

Publish and Subscribe Service defines how entities publish and subscribe to Platform Services. The Discovery service can be used to identify published services. This service is optional, but it becomes a mandatory service in scenarios where entities need to publish services and/or subscribe to Platform services.

[D12] Entities publishing services and subscribing to Platform Services **SHOULD** use the Publish and Subscribe Platform Service.

[R46] When entities need to publish services and/or subscribe to Platform Services a compliant platform **SHALL** make such service available.

4.6.3.12 Concurrency Platform Service

Concurrency Service defines how entities handle concurrency. Concurrency is the occurrence of different instances of events at the same time. Such events may or may not be dependent on each other. Concurrency may be allowed, banned or subject to certain restrictions depending on Policy and use case. An example for a banned concurrency would be for two entities adding a block to a PDL at the same time (which will create a fork). An example of an allowed concurrency may be collection of information from multiple sensors at the same time and writing such information to a table in a certain order (for example alphabetical, ascending) as prescribed by Policy.

[R24] A platform **SHALL** implement a Concurrency Service based on Policy.

4.6.3.13 Storage related services

4.6.3.13.1 Types of Storage Platform Services

Storage can be implemented and used by a PDL Platform in numerous ways. The following clauses define such storage services.

NOTE: The definition of "Directly Connected Storage" in the context of this clause is that the storage is local to the node or is external storage that is managed by the owner of that node. It includes internal RAM, internal file system, external drive, NAS and Cloud storage services. It is not limited to storage that is physically connected to a node.

4.6.3.13.2 In Memory Storage Platform Service

4.6.3.13.2.1 Definition of In Memory Storage

In Memory Storage is any data that is stored in the RAM (or RAM swap space on a local disc) of the computer running an application.

There are two types of In-Memory Storage options, see clauses below.

4.6.3.13.2.2 Volatile storage

The contents of volatile storage are not to be recorded anywhere and will not survive a restart of the computer or application.

[R25] Volatile in Memory Storage **SHALL NOT** keep a copy of the RAM contents on a disc or any other sort of non-volatile memory.

[R26] If a node uses RAM Swap Space for Volatile Storage the contents of such storage **SHALL** be erased when the node or application restarts.

4.6.3.13.2.3 Non-Volatile storage

The contents of non-volatile storage may be recorded on a local disc or non-volatile memory and may survive a restart of the application or computer.

[O23] Non-Volatile Storage **MAY** survive a node/application restart.

4.6.3.13.3 File System Storage Platform Service

Any storage on a directly connected storage such as a local disc, an external drive, a NAS or cloud storage.

[R50] Nodes **SHALL** support directly connected storage.

[O24] Devices used to access or run an application **MAY** support directly connected storage.

[R51] Access to NAS and Cloud storage **SHALL** follow all security and access policies prescribed by the governance.

4.6.3.13.4 On-Chain Storage Platform Service

On-Chain storage is any application data that is stored in blocks on all nodes using the chain. Each block in a chain is numbered and is identical to the respective blocks on all nodes using the chain. Data is stored locally on the node or on an external storage managed by the owner of the node.

[R52] Each On-Chain block **SHALL** have a unique number.

[R53] Each On-Chain block **SHALL** be identical to all other blocks carrying that unique number on other nodes participating in the chain.

[D13] A node **SHOULD** store the On-Chain blocks on directly connected storage that is physically connected to the node.

[O25] A node **MAY** store On-Chain blocks on directly connected storage that is not physically connected to the node if it is managed by the owner of the node and follows all security and access policies prescribed by the governance.

[R54] On-Chain storage **SHALL** be secured according to the security policy defined by the governance.

4.6.3.13.5 Off-Chain Storage Service

4.6.3.13.5.1 Definition

Off-chain data storage is the storing of information in a digital, machine-readable medium that is not stored on the main chain. The main differentiator between On-Chain and Off-Chain storage is that Off-Chain storage is not loaded as a block to the main chain used by all nodes.

Off-chain storage is a key enabler to scale blockchain-based applications that are data-intensive and/or data sensitive. It is often used to store non-transactional data that is too large to be stored in the blockchain efficiently or requires the ability to be changed or deleted. Off-Chain data is typically only accessible by a subset of the nodes participating in a chain.

There are two types of off-chain storage, see clauses below.

4.6.3.13.5.2 Distributed Addressable Storage

Distributed Addressable Storage (for example IPFS) is content that can be accessed through a link (URL). Such URL may be loaded into the main chain thus such off-chain storage is accessible by all nodes even though it is not loaded to the main chain. Addressable storage may be distributed (for example stored on a distributed ledger, with or without blockchain) which is then called "Distributed Addressable Storage".

4.6.3.13.5.3 Non-Addressable Storage

Non-Addressable Storage is content that cannot be addressed and accessed by any other entity except for the entity that directly manages this data.

[R55] Off-Chain data **SHALL** be accessible to the node to which it is directly connected.

[R56] Addressable Storage Off-Chain data **SHALL** be accessible by any other node meeting the access control policies defined by the owner of the node to which it is directly connected.

[O26] Addressable Storage Off-Chain data **MAY** be accessible by any other node meeting the access control policies defined by the governance.

[O27] Off-Chain data **MAY** be stored on a sidechain.

4.6.3.13.6 Distributed Blockchain Storage Platform Service

The concepts of Distribution, blocks and a chain are not necessarily interdependent. Data may be stored on a single location or may be distributed, regardless of the type of data (blockchain or other). On the other hand, blocks can be chained (using hashes or other methods) and stored locally on a non-distributed ledger.

The definitions in this clause apply to the specific case of a distributed blockchain ledger, that is, scenarios where the blockchain is stored in a distributed ledger. Such scenarios also include provisions to ensure integrity of the data across the distributed nodes.

The Distributed Blockchain Storage Platform Service is inherent to the PDL. Each PDL node stores the exact same copy of the chain as all other nodes in a PDL. However, situations may arise where certain nodes add invalid blocks thus invalidating the entire chain. Those are considered temporary events and the governance and consensus mechanisms offer ways to identify such invalid blocks/chains and to take actions to eliminate the problem. The methods by which such events are treated, and such situations are resolved vary by PDL type, consensus mechanism and governance and are beyond the scope of the present document.

[R27] Each node **SHALL** store the exact same chain as all other nodes on a PDL.

[R28] When a node detects an anomaly or a discrepancy between the chain stored on it and the consensus-driven chain stored on other nodes it **SHALL** flag its chain as "invalid" and replicate the entire chain, or the invalid parts of the chain, from a valid node holding a valid chain.

[R29] Upon replication of a valid chain from a valid node the node **SHALL** recalculate the hashes to ensure validity of the new blocks and upon successful recalculation it may remove the "invalid" flag.

Due to the structure of the chain as linked blocks, the validation of an invalid chain can be achieved by only replacing the blocks that include and follow the invalid block and any subsequent block or blocks added to the chain afterwards. Thus, it is not required that the entire chain is replaced.

4.6.3.14 Modelling Related Platform Services

4.6.3.14.1 Introduction to Modelling

The Modelling Platform Services define part of the common vocabulary and concepts for the platform. Those are:

a) Information Model.

b) Data Models.

Modelling Platform Services also define services that are required for using a model as a common vocabulary:

c) Model Search.

d) Model Stitching.

The Modelling Tier Platform Services consist of services that are fundamental for building more powerful PDL Services as well as distributing a platform. For any PDL platform the Information Model and the Data Model Platform Services are mandatory.

[R60] The Information Model Platform Service **SHALL** be implemented on all PDLs.

[R61] The Data Model Platform Service **SHALL** be implemented on all PDLs.

4.6.3.14.2 Information Model

An information model represents concepts of interest to the management environment in a *technology-neutral* form. An information model is a template and is not meant to be instantiated. Rather, data models derived from it are instantiated.

[R62] Implementations **SHALL** use a single information model to represent managed objects.

To accommodate future changes and applications a platform should be designed in an extensible manner so additional modules could be added to it to facilitate such applications.

[D14] Implementations **SHOULD** use a modular and extensible information model.

For example, an information model would define generic objects such as location, entities, ownership, device types, functionalities, and other high-level concepts. This information model could then be used for an abstract description of applications from different disciplines: Agriculture, Health, Telecoms, Weather, Financial services, etc.

[D15] Information models **SHOULD** be specified as a standard in a formal document issued by an SDO.

4.6.3.14.3 Data Model

A data model represents applicable concepts in an implementation in a *technology-specific concrete* form. Sample technologies could be Agriculture, Health, Telecoms, Weather forecasting, Financial services, etc.

An implementation may require multiple data models that represent objects using different repositories, protocols, and data formats. Data Models represent application specific, lifecycle-step specific and product specific implementations and are derived from the respective parts of the Information Model. Since all Managed Objects require at least one PDL Software Interface through which they can be managed, and all Software Interface communications require a Data Model that defines the representation of data exchanged through such interface, each Managed Object needs to have at least one Software Interface and at least one Data Model implemented through that interface.

[D16] Every Managed Object **SHOULD** be represented in at least one Data Model.

[O28] Every Managed Object **MAY** be represented in two or more Data Models.

[D17] Implementations **SHOULD** associate the software interfaces provided by a particular IRP with a set of class methods of a Data Model.

NOTE: Class methods may be invoked directly (for example using code) or indirectly (for example using APIs or DSLs).

[O29] Data Models **MAY** be Application Specific.

Each Data Model is derived from a single Information Model, which facilitates reconciling these different representations of the same concept into a single object. There are currently three common practices to structuring an Information Model:

a) **Bottom-Up:** Construction of an Information Model from multiple Data Models through iterative, consensus based, manual processes.

b) **Top-Down:** Design of a high-level, abstract, and modular, Information Model in a manner that allows adding up content and capabilities as required.

c) **Iterative Development:** use of bottom-up and top-down methods iteratively to produce the information model

These processes are typically performed through industry standard body meetings to construct an accepted Information Model. This process may be automated in the future through Model-Driven Engineering where an application (possibly embedded into the PDL) derives data models from the information model by crossing it with the specific object and functionality.

[R63] Data Models used in implementations **SHALL** be derived from the ETSI-ISG-PDL agreed information model.

[O30] Data models used in implementations **MAY** be derived automatically by the PDL platform.

[D18] Data models **SHOULD** be specified as a standard in a formal document issued by an SDO.

[O31] Platforms **MAY** adopt Data Models developed by other fora or SDOs.

[R64] Data Models Adopted from other fora or SDOs **SHALL** be compliant with an ETSI-ISG-PDL defined Information Model.

[R65] Data Models Adopted from other fora or SDOs **SHALL** be formally approved by ETSI-ISG-PDL.

An example of a data model, the concept of a "Tractor", refined from a higher-level concept of an "Agricultural Machine", would contain specifics about the engine, wheels, power ratings, transmission, manufacturer, and other factors. However certain attributes of a Tractor are shared with other machines (for example cars and tractors both have an engine and transmission) and some attributes may be specific to a tractor (for example the power-take-off mechanism that activates attached devices). These and other attributes are represented using inheritance and various software patterns.

4.6.3.14.4 Model Search

Model search is the functionality that allows a developer or an application to search for specific or generic models within existing information and data models. Such search functionality may assist a developer or an application in deciding what needs to be added to a model to support certain applications or application functionalities.

[R66] The model search functionality **SHALL** have full visibility to the Information Model and all Data Models in use by a platform.

[D19] The model search functionality **SHOULD** provide a human-readable response to queries based on keywords and text snippets (for example "tractor" or "machine" in the above example).

[O32] The model search functionality **MAY** offer a GUI based search in a model tree representation.

[D20] The model search functionality **SHOULD** provide API access to queries made through external search engines.

[R30] API access by external search engines **SHALL** be controlled by the governance.

4.6.3.14.5 Model Stitching

Model stitching is the functionality that enables integrating multiple models or parts of models into a single model. The resulting model shall be duplicate-free so if two models or parts of models each include the same objects or relations between objects - such duplicates are removed. If the models or parts thereof include objects of the same name that offer different purposes or relations, the resulting model shall separate those to unique objects with unique names and relations.

[R31] Stitched models **SHALL NOT** include duplicate attributes.

[R32] Each attribute **SHALL** be unique in a Stitched model.

4.6.3.15 Topology Platform Service

The Topology Platform Service allows a node to identify other nodes on the PDL Platform and, depending on consensus mechanism, identify which nodes to communicate with when performing PDL related tasks such as consensus and block replication. The number of nodes with which a node should communicate depends on the consensus mechanism, total number of nodes, number of valid nodes and governance.

[R70] The Topology Service **SHALL** publish the status of the node and the chain to all other nodes in a PDL.

[D21] The Topology Service **SHOULD** maintain the status of a sufficient number of nodes as required by the governance.

[O33] The number of nodes required to perform distributed PDL tasks **MAY** vary with time depending on number of valid nodes at any given moment.

The discovery process through which the Topology Service identifies other nodes depends on the specific governance and PDL type and is out of the scope of this clause.

4.6.3.16 Event Processing Platform Service

The Event Processing Platform Service processes events as they occur.

Such events are broken to different categories, and are presented here from the perspective of a specific node in a PDL:

a) Events that occurred locally on a specific node and do not affect other nodes nor do they affect the chain or consensus mechanism. For example, a user had logged in to the node or a backup of data was initiated. Such events are defined as *insignificant* for the proper function of the PDL.

b) Events that occurred locally on a specific node and may affect other nodes or the behaviour of the chain, including the consensus mechanism. For example, the storage device is reporting errors, CPU usage had reached a threshold, CPU is overheating, a block is validated, a block is invalid thus the node is flagged "invalid". Such events are defined as *significant* for the proper function of the PDL.

c) Events that occurred on other nodes and may affect the chain or the consensus mechanism. For example, a block is validated, a block is invalid, a specific node is flagged as "invalid", a specific node is in jeopardy (storage errors, CPU overheat, etc.). Such events are also defined as *significant* for the proper function of the PDL.

[D22] The Event Processing Platform Service **SHOULD** collect platform wide events.

[R71] The Event Processing Platform Service **SHALL** store the events On-Chain.

[R72] The Event Processing Platform Service **SHALL** process all Significant Events.

[O34] The Event Processing Platform Service **MAY** process Insignificant Events.

[R73] The Event Processing Platform Service **SHALL** notify the governance when Significant Events have caused a change to consensus operations.

[R74] PDL nodes **SHALL** follow governance on behaviour upon occurrence of Significant Events.

[O35] The Event Processing Platform **MAY** trigger actions independent of the governance, upon occurrence of certain Significant Events, based on smart contracts or prescribed lists of actions.

[R75] The Event Processing Platform Service **SHALL** trigger actions when specific events occur based on a prescribed list of actions.

[O36] The Event Processing Platform Service **MAY** use Artificial Intelligence when triggering actions based on events.

4.6.3.17 Distributed Data Collection Platform Service

The Distributed Data Collection Platform Service performs tasks related to collection of data. Data collection is defined in the following matrix.

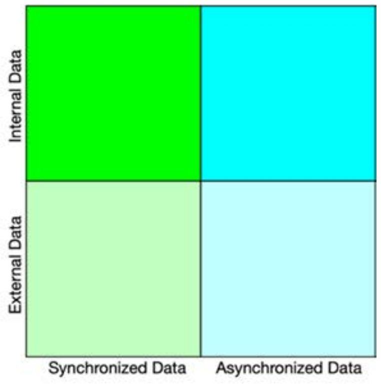

Figure 11: Data Collection matrix

Internal data is data that is generated by a node either through calculation or through a directly connected sensor (for example a thermometer) that feeds data to a specific node.

External data is data obtained from external resources or systems. For example, stock value or foreign exchange rates obtained from the stock exchange or bank.

Synchronized data is data that needs to be stored in synchronization with other data and thus requires sequencing and has dependency on timing or content of other data being collected. The methods used to ratify the integrity of synchronized data may vary depending on PDL and governance.

Asynchronized data is data that does not require synchronization and does not have dependency on other data and other data does not depend on it.

[R733] The governance **SHALL** define the level of synchronization required for data.

[O37] The level of synchronization **MAY** vary depending on application and type of data.

[R34] The governance **SHALL** define the method by which data is synchronized and the method of assuring synchronization meets such criteria.

[D23] The application and governance **SHOULD** define the type of data that needs to be collected and the duration it should be stored.

4.6.3.18 Distributed Secret Sharing Platform Service

Secret Sharing is sharing of confidential data between nodes in a manner that maintains confidentiality of the data. The method by which data remains confidential (for example encryption) is out of the scope of the present document.

[O38] A PDL **MAY** offer a Distributed Secret Sharing service.

[R35] When a Distributed Secret Sharing service is available it **SHALL** meet the confidentiality requirements defined by the governance.

[D24] The data shared through Secret Sharing does **NOT HAVE** to be stored on the chain.

4.6.3.19 Resource Management Platform Services

4.6.3.19.1 Introduction to Resource Management

The Resource Management Platform Services consist of a set of platform services that enable resources to be discovered, administered, managed, inventoried, composed, and virtualized.

4.6.3.19.2 Resource Discovery

The Resource Discovery Platform Service provide means to discover resources available to Platform applications and nodes.

For example, an application can discover resources available through platform services. Such resources can be computation power, storage space, connectivity between certain locations, sensor, or other equipment availability at certain locations.

[R36] A Platform resource **SHOULD** be discoverable.

[R80] The Resource Discovery Service **SHALL** support discovery of Platform resources.

4.6.3.19.3 Resource Virtualization

Resource Virtualization is the act of creating a virtual resource that mimics the behaviour of a physical resource. A virtual resource is constructed through software based on one or more physical devices. Such device(s) can be used to create one or more virtual resources that can be used by services and applications.

[R81] The Platform **SHALL** allow use of virtual resources.

[O39] A Platform Virtual Resource **MAY** be constructed using more than one physical resource.

[R82] A Platform Virtual Resource **SHALL** offer functionality that complies with the specifications of such resource.

4.6.3.19.4 Resource Inventory Management

4.6.3.19.4.1 Categories of Resource Inventory management

Resource Inventory Management is divided to two categories:

a) Node Specific Resources

b) Platform resources

[R83] The node management **SHALL** keep track of all resources available on that node, both those available to all platform users and those that are node specific.

4.6.3.19.4.2 Node-specific resources

Node specific resources are only available for use of the node on which such resource is installed. For example, directly connected storage that is not addressable.

[R84] Node-specific resources **SHALL** be discoverable and usable only by applications, services, and users of the specific node on which the resource is installed.

4.6.3.19.4.3 Platform resources

Platform resources are available to all nodes, services, applications, and users regardless of the node(s) on which it is implemented or installed. For example, an addressable IPFS storage or a network printer or a sensor.

[R85] The governance **SHALL** keep track of inventory and serviceability of all Platform resources.

[D25] Platform resources **SHOULD** be available to all Services, Applications, and users on any node.

[O40] Platform resources **MAY** be only available to select Services, Applications, and users on specific nodes.

4.6.3.19.5 Resource Administration and Management

Resource administration and management is an integral part of any platform. As described in previous sections, in a distributed platform resources may be associated with specific nodes, be accessible by specific nodes or be available and accessible by multiple, possibly all, nodes participating in the platform.

[R86] The Platform **SHALL** provide means to manage and administer Platform Resources.

[O41] The Platform **MAY** provide means to manage and administer Node-specific resources.

[R37] The Platform Resource administration and management service **SHALL** retain the exclusivity of Node-specific resources.

[R38] A node **SHALL** provide means to manage and administer Node-specific Resources.

4.6.3.19.6 Resource FCAPS

FCAPS is an acronym for *fault, configuration, accounting, performance, security,* which are the management tasks defined by the ISO model.

[D26] The Platform **SHOULD** offer FCAPS in accordance with the present document.

[R39] Platform resources **SHALL** support FCAPS functionality.

4.6.3.19.7 Resource Composition

Resources may be composed from other resources. Such resources are referred to as Composite Resources. As such their management and administration should follow the hierarchy of resources so that usage of a composite resource will mark the resources it is composed of as being in use. The same applies to all FCAPS functions.

[O42] Platform resources **MAY** be composed from other resources.

[R90] Composite Platform resources **SHALL** support FCAPS functionality.

4.6.3.20 Platform Service Management Platform Services

4.6.3.20.1 Introduction to Platform Service Management

The Platform Service Management services define how to discover, administer, and manage Services for a platform. They consist of a set of platform services that enable services to be discovered, administered, managed, inventoried, and virtualized. FCAPS operations, as well as the ability to compose new Services from existing Platform Services, are also included.

4.6.3.20.2 Platform Service Discovery Platform Service

The Platform Service Discovery Platform Service provides means to discover Platform Services available to Platform applications and nodes.

For example, an application can discover Platform Services available on a platform. Such Platform Services can be any of the Platform Services described in the present document as well as additional services that may be implemented on a platform by any party authorized to do so.

[R91] A Platform service **SHOULD** be discoverable.

[R92] The Service Discovery Platform Service **SHALL** support discovery of Platform Services.

4.6.3.20.3 Platform Service Virtualization

All services implemented through software can be considered as being virtual as there is no dedicated machine performing a service and it is done through code running on a processor and using resources of the node it is running on. For the purpose of the present document Platform Service Virtualization is the act of creating a Platform Service using virtual resources. A Virtual Platform Service is constructed through software based on one or more virtual resources.

[R93] The Platform **SHALL** allow use of Virtual Platform Services.

[O43] A Platform Virtual Platform Service **MAY** be constructed using one or more virtual resource.

[O44] A Platform Virtual Platform Service **MAY** be constructed using two or more other services where at least one of which is virtual.

NOTE: When a composite service is constructed of multiple objects (for example Resources, Services), none of which being virtual, it is considered a regular Composite Platform Service, not a Virtual Platform Service.

[R94] A Platform Virtual service **SHALL** offer functionality that complies with the specifications of such service.

4.6.3.20.4 Platform Service Inventory Management

A platform manages inventory of Platform Services through the governance that keeps track of inventory and serviceability of all Platform Services and manages availability of such Platform Services to applications and users.

[D27] The governance **SHALL** keep track of inventory and serviceability of all Platform services.

[D28] Platform services **SHOULD** be available to all Applications and users on any node.

[O45] Platform services **MAY** be only available to select Applications and users on specific nodes.

4.6.3.20.5 Platform Service Administration and Management

Service administration and management is an integral part of any platform. In a PDL, management tasks are handled through governance.

[R95] The Platform **SHALL** provide means to manage and administer Platform Services.

[R96] All Platform Services implemented on a platform **SHALL** be authorized by the governance.

[R40] Users **SHALL NOT** implement Platform Services without permission from the governance.

4.6.3.20.6 Platform Service FCAPS

FCAPS is an acronym for *fault, configuration, accounting, performance, security,* which are the management tasks defined by the ISO model.

[R98] Platform Services **SHALL** support FCAPS functionality.

4.6.3.20.7 Platform Service Composition

Platform Services may be composed from other Platform Services. Such Platform Services are referred to as Composite Platform Services. As such their management and administration should follow the hierarchy of Platform Services so that usage of a Composite Platform Service will mark the resources used by the Platform Services it is composed of as being in use. The same applies to all FCAPS functions.

[O46] Platform services **MAY** be composed from other services.

[R41] Composite Platform services **SHALL** support FCAPS functionality.

4.6.3.21 Application Management Services

4.6.3.21.1 Introduction to Application Management

The Application Management Services are a set of Platform Services that enable Platform Services to be composed and orchestrated into an application. In addition, resources that are used to support such Platform Services can also be orchestrated.

4.6.3.21.2 Application Composition

As described earlier in the present document, service composition is the act of composing a service from two or more other services. When Applications are concerned, they too can be composed of other applications or re-use other applications. For example, a weather broadcasting application can be composed of a weather forecasting application and data distribution application combined such that the application user can tailor the weather forecast and timing of distribution. An application may also be composed of a mix of applications and services.

[R100] The platform **SHALL** support Application and Service composition.

[R101] The platform **SHALL** provide Application, Service and Resource management services to composite applications.

4.6.3.21.3 Application and Platform Service Orchestration

When objects (Applications and Platform Services) are combined into a composite object there may be a need to orchestrate the construction and behaviour of the composite object. For example, when an application of which a composite object is constructed is using the output of another application as its input, the applications should be orchestrated such that the data traverses the applications in the right sequence.

[R102] The governance **SHALL** ensure data traverses the applications and services of which a composite object is constructed in the appropriate sequence as designed by the developer of the composite object.

4.6.3.21.4 Orchestration Platform Service

Composite objects require Orchestration to operate correctly. Orchestration is the act of chaining the objects in a manner that connects the respective ingress and egress interfaces of such objects in a topology and sequence that yields the required functionality. The Orchestration service provides the necessary management tools to chain the objects, publish them and manage their composite operation.

[R103] The Orchestration Platform Service **SHALL** chain objects as defined and designed by the governance.

[R104] The Orchestration Service **SHALL** apply all Resource, Service and Application management requirements as defined in the previous sections on the resulting composite object.

4.6.3.21.5 Platform exploration

Platform exploration is a functionality that allows an application to indicate its requirements and explore whether the platform offers such service capabilities.

[R105] A platform **SHALL** allow applications to explore its capabilities by indicating requirements and identifying Platform Services that may address such requirements.

4.6.3.21.6 Application Registration

Application registration is a functionality that registers and lists all applications operated on a platform.

[R106] A platform **SHALL** maintain a list of all applications registered and operated on it.

4.6.3.22 Transaction Management Service

Transaction Management Service (TMS) facilitates transaction related interactions between applications/services and underlying PDL (or PDLs) by providing the following functionalities:

a) Configure transaction-related policy rules for and/or to applications/services.

b) Receive and authenticate transaction-related requests (for example a request for creating a transaction) from applications/services.

c) Select an appropriate underlying PDL network for applications/services in scenarios where such a selection exists (for example a platform that handles multiple PDL networks).

d) Interact with underlying PDL networks and/or external storage on behalf of application and services, to retrieve and send transaction-related requests and data to and from applications and services; This may include:

1) retrieve transaction-related data from external data sources for applications/services.

2) receive responses from underlying PDL networks and/or external storage; and

3) process and forward the responses to applications and services.

[R42] A Transaction Management Service **SHALL** authenticate, process, and manage incoming transaction operations from applications and services.

[R43] A Transaction Management Service **SHALL** interact with designated underlying PDL networks and/or external storage to fulfil transaction operations on behalf of applications and services.

[R44] A Transaction Management Service **SHALL** process responses for transaction operations as received from designated underlying PDL networks and/or external storage and forward them to applications and services.

[O47] A Transaction Management Service **MAY** configure transaction-related policy rules for applications and services.

[O48] In a platform that handles more than one PDL network, a Transaction Management Service **MAY** select an underlying PDL network to be used for transactional purposes by applications and services.

[O49] In a platform that handles or has access to external storage, a Transaction Management Service **MAY** handle transaction related activities using such external storage for applications and services.

4.6.3.23 Data Model Gateway/Broker

4.6.3.23.1 Introduction to presentation services

Each product or Functional Block may have its own Data Model, and possibly one or more interfaces through which it exchanges data with other entities. When different entities that use different Data, Models need to exchange information a Data Model Broker, also called a Data Model Gateway, is required to ensure information is exchanged correctly. Such Broker/Gateway is software that mediates between two systems with different data models yet enabling the two different systems to communicate transparently with each other. There are many benefits of using Data Model Brokers, including error reduction via software transmitting data instead of humans and business process automation through automated transfer of data between applications. Data Model Brokers also enable custom applications that integrate different application data.

The purpose of the Data Model Broker/Gateway is to:

a) translate data communicated from an external system/entity into a normalized form that all platform Functional Blocks can understand; and

b) translate recommendations and commands from the normalized form of a platform to a form that the external system/entity can understand; and

c) manage authentication, accounting, and authorization of the entities that want to communicate with a platform.

As discussed earlier in the present document, APIs are the most common method of data exchange between entities and Functional Blocks, hence an implementation of a Data Model Broker/Gateway in an environment where APIs are in use will be in the form of an API Broker/Gateway.

An API Broker ingests APIs through an appropriate Reference Point, analyses the API, and then routes the functionality of the ingested API to an appropriate Platform Functional Block. Similarly, APIs sent to external clients are sent to the API Broker, which routes the functionality of the API to the appropriate client.

Alternative information exchange methods exist and may be used between a platform and external entities or between objects such as with use of micro-services. The data model broker in such instances will offer the same functionality as the API Gateway but will use communication methods compliant with such objects. One such example is the use of "hot-folders" which are IPFS where data can be exchanged by uploading/downloading data files by multiple entities using a file transfer method such as SFTP. The use of such alternative data transfer methods may still require brokering between data models/formats. While "hot folders" may be easier and faster to implement than APIs they are typically less secure than APIs and brokering data models using such "hot folders" is more complex to implement than an API Broker/Gateway. The respective platform parties may choose an implementation that meets their requirements.

[R45] A platform **SHALL** exchange data with external entities and users through a Data Model Broker/Gateway.

4.6.3.23.2 API Presentation Platform Service

4.6.3.23.2.1 Introduction to APIs

An API is a set of communication protocols, code, and tools that enable one set of software components to interact with either a human or a different set of software components. APIs are critical for platform and ecosystem development. Effective API programs lay the foundations for digital transformation by enabling organizations to build a platform and develop an ecosystem.

4.6.3.23.2.2 RESTful-APIs

A REST API (also known as RESTful API) is an API that conforms to the constraints of REST architectural style and allows for interaction with RESTful web services. REST stands for *representational state transfer*. The definition of REST and REST compliance is beyond the scope of the present document.

4.6.3.23.2.3 Non-RESTful-APIs

APIs that do not conform to the REST architecture style are considered Non-RESTful.

[D29] All platforms and services **SHOULD** use RESTful-APIs.

[O50] APIs **MAY** be non-RESTful.

4.6.3.23.3 Micro-services

Micro-services may exchange information with other objects without an API. In such case their functionality will be independent of the information contained in this clause.

4.6.3.23.4 Webhooks

Webhooks are triggered by events and are typically executed as an HTTP POST request by an originating functional block where such event had occurred (e.g. a Platform Service) to one or more target functional entities (e.g. another Platform Service, an external entity, an application, a PDL chain). The HTTP POST request is sent to a URL on the receiving functional block configured to receive such webhook. The technical definitions, security and authentication aspects of webhooks are beyond the scope of the present document.

4.6.3.24 Application Specific Services

As their name implies, Application-Specific Services serve a specific application or a group of applications that require this specific service but is not required by all applications operated using the platform. The characteristics of an Application-Specific service are that:

a) It is only used by Applications and cannot be used by other Platform Services.

b) It can be implemented as part of an application rather than as a Platform Service.

[R111] An Application Specific Service **SHALL NOT** be used by other Platform Services.

[O51] An Application Specific Service **MAY** use other Platform Services.

[O52] An Application Specific Service **MAY** be implemented as part of an application.

Due to their circumstantial nature the present document does not list such specific services. During the evolution of a platform - some application developers may consider moving certain functionalities from the applications they are developing to the platform thus making them available to other applications to use.

4.6.3.25 Accounting Service

The Accounting Service measures the consumption of resources and services by users and applications and generates a detailed usage report. The level of details in such report is to be determined by the governance and business requirements being supported. The commercial implications of using the platform by users may be based, in part or in whole, on such report.

4.7 Application Clients

4.7.1 Introduction to Application Clients

Application clients are the user front end that allows the user to interface with an application and perform application tasks such as initiating a transaction, performing a query, participating in a consensus vote, etc.

There may be multiple application clients to the same application, which differ in terms of the hardware and software implementation of the underlying device being used for the application client.

Some common types of application clients are listed herewith.

4.7.2 Computer Applications

Computer applications are running on a personal computer. There are multiple types of personal computers in common use today, notably the Microsoft Windows enabled PC (typically using an Intel or similar processor), the Apple Mac (which runs on both Intel and Apple processors), Linux®, the Chrome devices, and numerous others.

NOTE: Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries.

An application SHALL be tailored to the specific hardware and operating system that personal computer is using and offer a uniform interface to the user regardless of that operating system.

[D30] A Computer Application **SHOULD** offer functionality in a manner agnostic to the underlying hardware and operating system of the computer it is implemented on.

[O53] A Computer Application **MAY** require specific minimum hardware and software configurations to function properly.

[O54] A computer running an application **MAY** also be used as a network node.

4.7.3 Mobile Device Application

Mobile Applications run on mobile devices such as smartphones and tablets. Such mobile devices may run on various hardware types and vary in terms of operating system, screen size, computation power and internal architecture. While there is a challenge to maintain software compatibility with multiple mobile environments the benefit is the wide-spread adoption and availability of such devices which makes the application available to larger audiences. An additional benefit is the portability of mobile devices that makes a mobile device application available in locations where a personal computer cannot be operated or cannot be connected to the network.

Since mobile devices may be limited in resources, computation power and storage space, and would typically not have directly connected storage, they would not typically be used as network nodes. Furthermore, the application interface may lack some functionality that may only be available on other application types.

[R46] A Mobile Device Application **SHALL** offer sufficient functionality as required by the PDL application to perform tasks required by its user.

[O55] A Mobile Device Application **MAY** offer reduced functionality compared to other Device application types.

4.7.4 PDL Cloud Applications

Cloud Applications run on a network-based machine (typically a virtual machine) and are accessible through the Public Internet, through private networks or through Intranets, depending on the network environment implementation. Such applications would typically be accessed through an HTML GUI (for example web browser) using HTTP or HTTPS or other mark-up languages as used by the respective developers. The GUI would typically offer access to most, or all, application features.

The GUI implementation may vary depending on the Web browser used and the operating system of the device used to run such web browser.

[D31] A PDL Cloud application **SHOULD** be compatible with all commonly used Web Browsers on all commonly used operating systems as prescribed by the governance.

The definition of "commonly used" in the context of this requirement may vary with time, and is beyond the scope of the present document, thus the present document does not list the specifics and leaves such decision to the governance and developers of each specific application or platform.

[R47] A Cloud Application **SHALL** use a secure connection (for example HTTPS) to the web client.

[R48] A Cloud Application **SHALL** be compatible with a list of web browsers and operating system environments defined by the governance for each such application.

5 Summary

The PDL Reference Architecture defined in the present document offers an abstract architecture and an extensive list of Platform Services. To realize a platform the specifics, have to be designed and defined through implementation agreements and through adoption of existing implementations of services that can be made compliant with the requirements set forth in the present document. While being high level and abstract, the present document is all encompassing in the sense that it includes services that may be required in a very large list of use cases and environments. It is designed as general guidelines to be followed when defining the specifics, yet keeps the door open for future expansion, without prescribing the specifics for one use-case or another.

Annex A (informative): Change history

Date	Version	Information about changes
April 2022	V1.1.1	Publication
March 2023	V1.2.1	Revision

History

Document history		
V1.1.1	May 2022	Publication
V1.2.1	June 2023	Publication