ETSI GR PDL 003 V1.1.1 (2020-12)

Permissioned Distributed Ledger (PDL); Application Scenarios

Disclaimer

The present document has been produced and approved by the Permissioned Distributed Ledger 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
DGR/PDL-003_App_scenarios

Keywords
ledger, use case

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 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Contents

List of Figures

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is 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 IPR Policy, no investigation, 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.

Foreword

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

Modal verbs terminology

In the present document "**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.

1 Scope

The present document describes Permissioned Distributed Ledger Application Scenarios. The aim is to consider and describe the potential application scenarios for the operation of PDLs, including provision models with special emphasis on as-a-service paradigms, and PDL infrastructure governance aspects. The present document provides definition of terms to be used in the scenarios and recommendations for future normative specifications.

2 References

2.1 Normative references

Normative references are not applicable in the present document.

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] ISO TC307: "Blockchain and distributed ledger technologies".

[i.2] ISO 22739: "Blockchain and distributed ledger technologies -- Vocabulary".

[i.3] Recommendation ITU-T SG13: "Future networks including cloud computing, mobile and next-generation networks".

[i.4] ETSI GR PDL 001: "Permissioned Distributed Ledger (PDL); Landscape of Standards and Technologies".

[i.5] ITU-T Technical Specification FG DLT D1.1: "Distributed ledger technology terms and definitions", Definition number 6.22 "Fork".

[i.6] ITU-T Technical Specification FG DLT D1.1: "Distributed ledger technology terms and definitions", Definition number 6.25 "Hard Fork".

[i.7] World Economic Forum (04/2020): "Inclusive Deployment of Blockchain for Supply Chains: Part 6 - A Framework for Blockchain Interoperability".

NOTE: Available at https://www.weforum.org/whitepapers/inclusive-deployment-of-blockchain-for-supply-chains-part-6-a-framework-for-blockchain-interoperability.

[i.8] Jennifer J.Xu.:"Are blockchains immune to all malicious attacks?".

NOTE: Available at https://jfin-swufe.springeropen.com/track/pdf/10.1186/s40854-016-0046-5.

[i.9] Aljosha Judmayer, Nicholas Stifter, Alexei Zamyatin, Itay Tsabary, Ittay Eyal, Peter Gaži, Sarah Meiklejohn, Edgar Weippl: "Pay-To-Win: Cheap, Crowdfundable, Cross-chain Incentive Manipulation Attacks on Cryptocurrencies".

NOTE: Available at https://eprint.iacr.org/2019/775.pdf.

[i.10] ETSI GR PDL 004: "PDL: Smart Contracts Permissioned Distributed Ledgers System Architecture and Functional Specification".

[i.11] ETSI GR PDL 006: "Permissioned Distributed Ledger (PDL); Inter-Ledger Interoperability".

3 Definition of terms, symbols and abbreviations

3.1 Terms

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

API Gateway: API management tool that sits between a client and a collection of backend services and acts as a reverse proxy to accept all Application Programming Interface (API) calls, aggregate the various services required to fulfil them, and return the appropriate result

NOTE: See Red Hat® at https://www.redhat.com/.

chain: collection of PDL records concatenated to each other through a unique format and a specific order that may vary depending on PDL type

common functionalities: such functionalities that offer similar or same behaviour across multiple applications

core functionalities: such functionalities that exist in all applications

fork: occurence where a single blockchain is split into two or more separate blockchains by having different blocks appended to it. Once Forked, each blockchain becomes an individual Chain

node: device (real or virtual) that transacts on a Chain

omni-Lateral: PDL that all (Omni) nodes in a platform participate in

PDL Platform: collection of Nodes transacting in a coordinated manner on one or more Chains

PDL Type: PDL variations by Functional components such as Chain structure, Consensus mechanism, Publicity, Implementation, Security, Discoverability

Permissioned Distributed Ledger (PDL): distributed Ledger that maintains access control to allow certain actions to be performed only by certain identifiable participants

NOTE: See clause 4.1 of the present document.

transacting node: node that transacts on a PDL but does not participate in a Consensus Vote

validating node: node that participates in a Consensus vote

3.2 Symbols

Void.

3.3 Abbreviations

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

AML	Anti-Money Laundering
API	Application Programmable Interface
AS	Autonomous System
BGP	Border Gateway Protocol
CPU	Central Processing Unit
DLT	Distributed Ledger Technology
DNS	Domain Name Service
ETSI	European Telecommunications Standards Institute
EU	European Union
GDPR	General Data Protection Regulation

GPS	Global Positioning Service
GR	Group Report
GUI	Graphical User Interface
HR	Human Resources
ICT	Information and Communications Technology
IP	Internet Protocol
ISG	Interim Study Group
ISO	International Organization for Standardization
IT	Information Technology
ITU	International Telecommunication Unit
ITU-T	ITU Telecommunication standardization sector
KYC	Know Your Customer
MVP	Minimum Viable Product
NAS	Network Attached Storage
OS	Operating System
OSI	Open Systems Interconnection
P2P	Peer-to-Peer
PDL	Permissioned Distributed Ledger
RAID	Redundant Array of Independent Disks
RAM	Random Access Memory
SATA	Serial Advanced Technology Attachment
SHA	Secure Hash Algorithm
SLA	Service Level Agreement
SMS	Short Message Service
SWIFT	Society for Worldwide Interbank Financial Telecommunication
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TPS	Transactions Per Second
VM	Virtual Machine
WEF	World Economic Forum
ZKP	Zero Knowledge Proof

4 PDL Reference Framework

4.1 Introduction

Chains differ from one another by multiple dimensions, such as consensus mechanism, transaction rate, programmability (through smart contracts), tamper-immunity and many others. The present document does not intend to compare different chain types nor is it the intention here to make a recommendation of a specific, or group, of chains. The commonalities are that PDL platforms can be Manageable and Governable, have to be Accessible by Applications and Services through APIs, and could, depending on DLT type, be programmable through Templates (that may be dependent on chain type).

4.2 Definition of PDL

PDL is a Distributed Ledger architecture offering modular possibilities with permission based processes.

Permissioned Distributed Ledger are commonly divided into two different approaches:

a) Public Permissioned Distributed Ledger.

b) Private Permissioned Distributed Ledger.

The operational scenarios of such ledgers are discussed in greater detail in clause 6.2.3 of the present document and can be generally defined as follows:

- Public PDL scenarios operate with no restrictions once an acceptance process is completed. Such process is part of the governance and operation of a DLT, which also include other policies with the aim to provide trustworthiness and a public or general-purpose service.

- Private PDL scenarios allow access to select members that have passed a certain selection criteria defined through the governance structure, and are aimed for a private or a specific purpose service.

4.3 Abstract PDL Architecture

In order to provide a basis for discussing various PDL Application Scenarios, an abstract, simplified, reference framework and functional diagram is used. Figure 1 herewith defines the three abstract layers that appear in most PDL implementations. They are discussed here one by one, as well as the relations and dependencies therein.

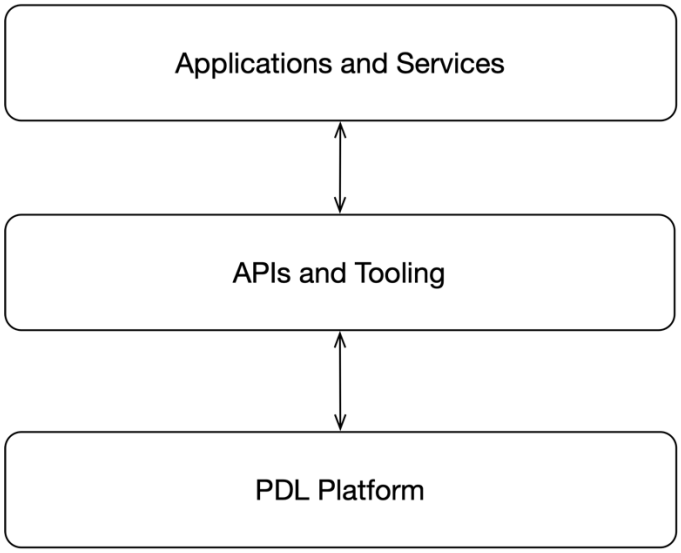

Figure 1: Abstract PDL Reference Framework

The Applications and Services layer represents the actual customer/consumer facing application, be that consumer an individual accessing the PDL platform through a portal or an application, or another platform/device accessing the PDL platform through an API or other means of electronic transactions. Depending on PDL type, an application can be implemented as a smart contract, as an external code/application (which may be platform or operating system specific e.g. mobile phone, desktop computer, IoT device) or as a mix of both.

It is beyond the scope of the present document to discuss the applications themselves and the methods by which they interface (northbound) with the customer/consumer.

Applications would require exchange of information with the PDL Platform itself, possibly more than one PDL, through a southbound interface. Such PDLs may be developed and operated by different entities.

APIs and Tooling layer allows interaction between the applications/services and the PDLs. This layer is referenced by ISO as the ***API Layer.*** For consistency, the present document follows the term used by the EU Blockchain Observatory and Forum. This layer allows abstraction of the PDL Platform and Application layers in a manner that may allow applications to operate on more than one PDL Type and vice-versa. This layer consists of consoles, dashboards, and development environments made available to developers, institutional users, auditors and regulators.

The ***PDL Platform layer*** contains the PDL nodes as well as smart contracts, management and governance tools, and other software elements that are embedded into code running on the PDL nodes. The PDL platform may use any of the multitude of PDL types available at the time and may use governance and management tools that are compatible/interoperable with said PDL Type.

In certain scenarios, specifically when the application is embedded into the PDL platform (e.g. as a smart-contract) and the users interface the PDL Platform layer directly, the Application and/or API layers may not be required.

Examples would be:

1) Scheduled actions taken without external intervention occur on the PDL Platform layer and do not require the API and Application layers.

2) A smart contract in a PDL platform that can be accessed through an API through a third party application (e.g. wallet) that is not part of the PDL platform itself.

4.4 PDL Platform Governance and Management

The Abstract PDL platform architecture discussed in clause 4.3 is enhanced by additional Governance and Management functionalities as depicted in Figure 2 herewith.

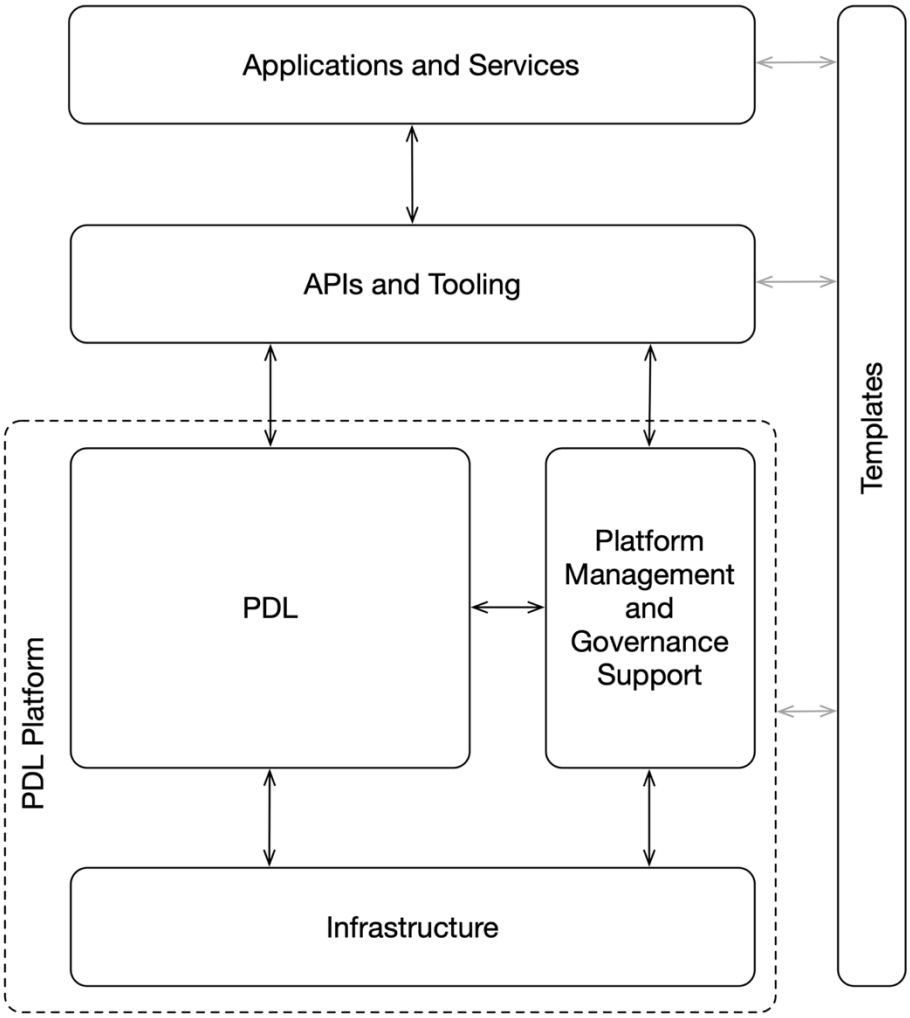

Figure 2: Simplified Functional PDL Reference Framework

The major difference between the abstract diagrams depicted in Figure 1 and Figure 2 is the separation of the Infrastructure from the PDL Platform, and the addition of a Management and Governance Support functionality.

The Platform is managed and governed through various methods, that may vary by PDL type, consensus mechanism and application.

The Platform Management and Governance Support functionality allows the continued operation of the entire platform by ensuring resources are available, governance requirements are met, and the PDL is operating in good order (e.g. no forks, a sufficient number of validating nodes are in operation, etc.). Governance can be implemented in multiple forms. In certain scenarios it can be implement through a hierarchical structure where a top-level entity governs the behaviour of the lower levels. In other scenarios there may be no hierarchy and governance is applied through consensus between equal-level entities. There are also hybrid models in existence where a human board of elected or appointed directors defines the rules of operation and the governance structure implements the same. This is discussed in greater details in clause 6.3 herewith.

***The Infrastructure*:** PDL platforms require computing, storage and communication infrastructure. The computing resources would typically be off-the-shelf blades, and in most cases can be implemented as VMs or containers in private or public clouds. It would be worth mentioning here that certain permissionless distributed ledgers require that the node is owned, operated and in-house by its user. This is not necessarily the case for permissioned distributed ledgers that the present document is about. Such nodes can, subject to regulations and respective policies, be operated in public cloud. Storage requirements depend on the choice of PDL, as well as the requirements of the applications. PDL platforms today typically use best-effort networks such as the public Internet as the means of communication between nodes and between users to the PDL platform. Future scenarios may introduce the need for managed communications between nodes in order to ensure predictable and secured performance.

Templates: Certain core and common functionalities, such as Identity, smart contracts, assurance, are defined as Templates that may be re-used and integrated with repetition in multiple instances and aspects. Templates are ready-to-use PDL implementations or functionalities that can be applied as needed. The may be implemented in multiple layers. This is a useful functionality that may serve, in certain scenarios, as a presumption of conformity with certain guidelines, requirements or specifications. Templates are not specific to a PDL layer. Examples of templates would be: Cyber-Security Type approval for a vehicle, Security and Privacy specifications for data storage, Commercial Reconciliation rules for bilateral settlement. The Identity of a person or an entity may be used in multiple applications and will preferably be presented in a common way regardless of the application and ledger. Templates are discussed in further details in clause 7.2 herewith.

4.5 Stakeholders

A PDL application may involve one or more of the below Stakeholders:

- End Users:
  - Humans. Individuals or groups of people that use a PDL application.
  - Machines (e.g. an application accessing a PDL without human intervention).
- Platform Operators:
  - PDL. A PDL can operate autonomously during certain phases of its lifecycle.
  - Management (e.g. governance, board). A PDL may be managed trough an internal/external governance structure.
  - Auditor/Monitor. A PDL may be monitored and audited for puposes.
- Software Vendors:
  - PDL developers. Software vendors who develop PDLs.
  - Application developers. Software vendors who develop applications that use PDLs.
- Infrastructure vendors:
  - Cloud. Vendors providing cloud infrastructure used to host PDL nodes or store PDL data.
  - Connectivity:
    - Fixed. Operators providing fixed/physical infrastructure used to convey information between PDL nodes, applications and users.
    - Mobile. Operators providing wireless/mobile infrastructure used to convey information between PDL nodes, applications and users.
- Regulatory and Governance Authorities:
  - Regulator. An entity or individual that regulates the behavior of a PDL during its lifecycle.
  - Legislator. An entity or individual that defines the rules and legal terms that a PDL applies to.
  - Auditor/Monitor. An entity or individual that Audits/Monitors the behaviour and activity of a PDL and indicates if a PDL is operating normally or not.

  - Authenticator. An individual or entity that authenticates validity of actions or other entities/individuals.

- Business Ownership and management:

  - Standard bodies. Industry associations or collaborative groups that are recognized as standard defining organizations.

Figure 3 herewith illustrates an evolved approach to a PDL platform that is agnostic to PDL types, application types, and thus can be the basis for an open eco-system. Certain application scenarios require PDL-type agnosticism. Such agnosticism applies to any unique chain that is subject to different operational scenario than other chains (e.g. different PDLs based on Ethereum chains will be considered different PDL Types). That would typically be the case in eco-systems where multiple, sometimes competing, entities use PDL in a federated wholesale supply chain. It may be impossible, or operationally/commercially non-plausible, to enforce the use of a specific PDL Type for an application. As depicted in Figure 3, the Distributed Ledger Platform may consist of multiple PDLs which are managed via the functionality of the Platform Management and Governance Support layer and synchronized in a manner that allows PDL-Type-agnostic applications to operate. Depending on the PDL types involved the Platform Management and Governance functionality may be broken down to sub-elements each managing a specific PDL type. Certain scenarios may also include PDL hierarchy (e.g. one PDL encompassing multiple PDLs). The Platform Management and Governance would then typically be structured in a manner that supports such scenarios.

User access to the PDL platform may be performed, depending on PDL type and design, either directly to the PDL via an API or through an external application.

Certain eco-systems, such as supply chain management platforms, may require a Distributed Ledger Platform that supports simultaneous use of both bilateral PDLs and Multi-Lateral/Omni-Lateral PDLs (also referred to as *Shared Ledgers* [i.1] and [i.2]) in tandem. While it is operationally and contractually simple to require that both parties in a bilateral arrangement use the same type of PDL, it may be complex (and in certain cases impossible) to enforce the use of a specific PDL in a multilateral environment where multiple entities, often competitors, operate. In such cases it is the task of the Platform Management Support function to enable inter-PDL-Type and inter-chain (in cases of multiple chains of the same PDL type) interoperability. This includes periodical synchronization between PDLs and chains at a frequency that meets the requirements of the respective applications. The operational overheads and required performance of such platform requires careful planning and sizing to ensure continuous operation.

The API and Tooling Abstraction layer in Figure 3 herewith differs from the *APIs and tooling Layer* shown in Figure 2 by adding an abstraction functionality that allows applications to interoperate with multiple underlying PDL type. In an ideal world any application will be able to operate on any PDL type, but in reality, there may be certain functionalities required by certain applications (e.g. high TPS rates) or certain security requirements that dictate the use of specific PDL types. This can be resolved through the use of an Application abstraction layer that allows multiple vendors to develop compatible and interoperable applications giving users choice of supply and PDL type. The templates mentioned earlier may serve as common functions on the application layer as well.

Specific functionalities, such as e.g. certification verification, conformance with regulations, can be supported by the architecture and can be implemented as a template, a smart contract, as an element in an application or as a mix of both.

4.6 PDL Platform Abstraction Layers

Most existing PDL implementations are packaged as a bundle that includes all layers. For certain applications this is the preferred approach for various operation, IP or security reasons. Looking ahead there may be a growing number of PDL implementations that abstract the application from the underlying PDL and allow implementing an application on multiple PDL types. It is obvious that a specific implementation by a specific group of entities will use the same PDL Type across all participants, however a similar implementation of that same application by another group of entities may use a different PDL type.

The Entities in Figure 3 and the Application, Vendor and Service provider are discussed in further detail in clauses 6 and 7.

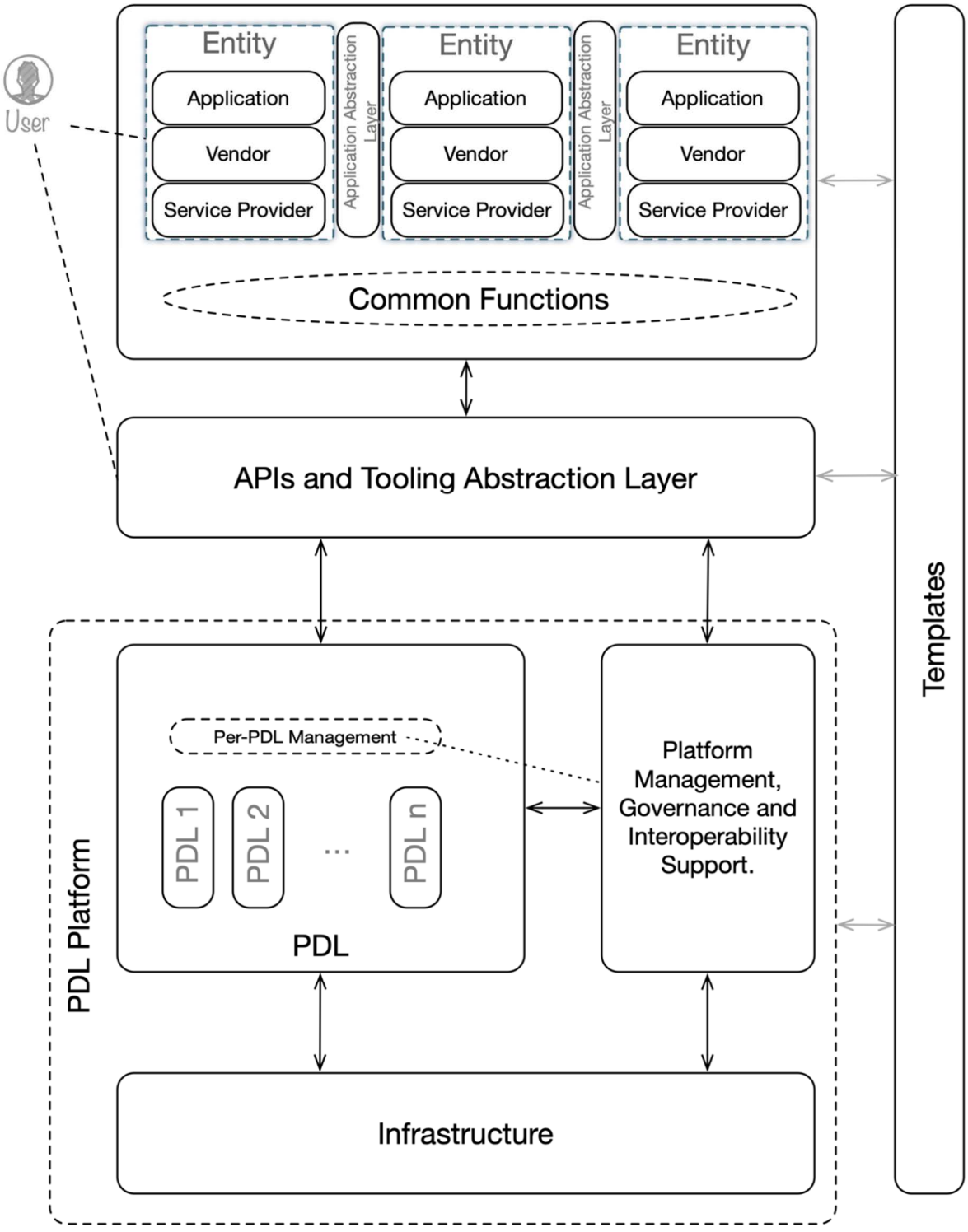

Figure 3: PDL Reference Framework

5 The infrastructure Layer

5.1 ICT Verticals

The PDL Infrastructure is built on three vertical families of components. The following clauses discuss the three families of components. Those verticals are depicted in Figure 4 . It is beyond the scope of the present document to discuss the commercial and operational aspects of implementation and integration of those components into a PDL.

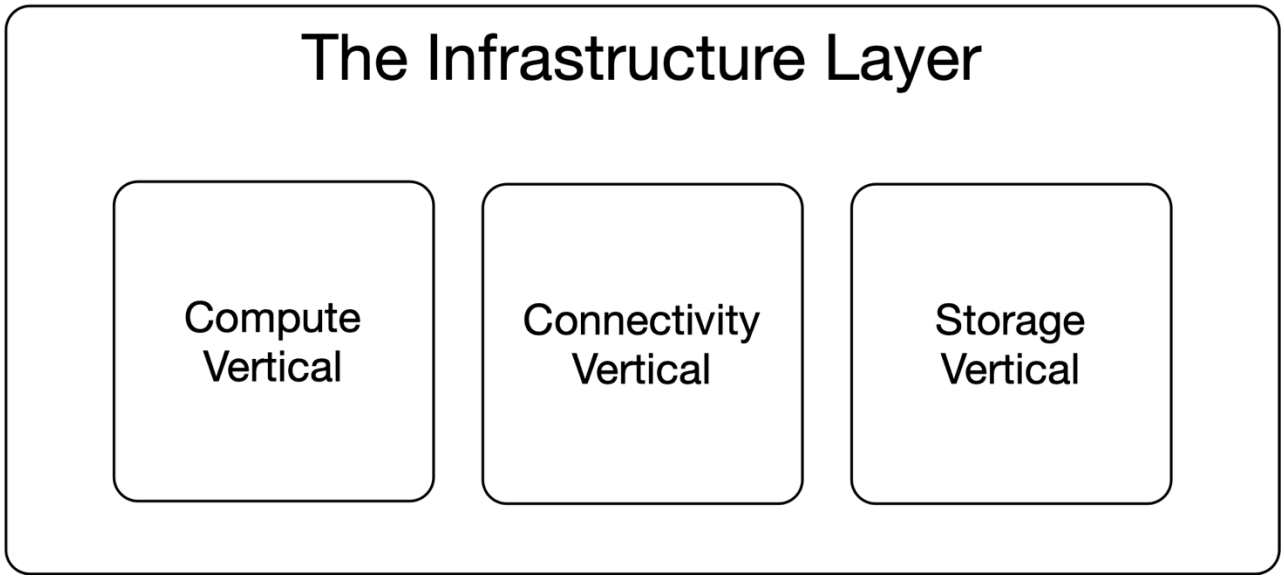

Figure 4: The Infrastructure Layer

5.2 The Connectivity Vertical

5.2.1 Introduction to the Connectivity Vertical

Connectivity provides the ability to move data from place to place. The nodes participating a PDL are typically connected with a Peer-to-Peer (P2P) network over IP protocol. In its simplest and easiest form one may choose to use the public internet, though this medium raises performance and security concerns. Dedicated connectivity options (such as leased lines or quality-managed VPNs) offer a more secure and predictable alternative but introduce commercial and operational overheads.

There are some aspects that the connectivity infrastructure should provide to the ledger.

5.2.2 Accessibility

Once a node is allowed to the ledger (e.g. via access control), the node is able to discover and connect to other nodes.

5.2.3 Availability

The nodes participating a ledger may locate in one cloud environment (e.g. in a same data center or even in a same local area network), or reside remotely from each other and connect with each other through wide area network. In either case, the P2P network needs to guarantee the availability for the connectivity.

5.2.4 Integrity

The connectivity infrastructure may encounter malicious behavior, e.g. BGP hijack, DNS hijack and TCP hijack. The infrastructure needs to continue guaranteeing the integrity of the communication data under these attacks or faults.

5.2.5 Confidentiality

The connectivity may encounter interception. The communication data should be kept confidential under interception.

5.2.6 Quality

Some application scenarios may require high-bandwidth and low-latency connectivity, to accelerate consensus or enhancing throughput of the ledger.

To this end, the connectivity can use secure end-to-end protocols (e.g. IPSec, TLS) for endpoint authentication, data integrity and confidentiality. A secure key-exchange mechanism is required in advance. The P2P network should provide robust connectivity, and infrastructure may need to defend against hijack and DDoS attacks and protect from network faults. Resource reservation may be used for quality of service.

5.3 The Compute Vertical

5.3.1 Introduction to the Compute Vertical

Compute is the act of performing computations on data. That would be the PDL Nodes and other devices performing management and governance tasks. Computations require a computing device, either a dedicated computer or a virtual resource such as a VM or a container in a cloud.

5.3.2 Jurisdiction

Depending on application and regulation the physical location of a node may be restricted to certain jurisdictions. E.g. Certain countries demand that nodes that process personal information of its citizens be located within the geographical borders of that country.

5.3.3 Privacy and Confidentiality

Certain entities (e.g. Banks) would only allow nodes to operate within their own facilities and will not allow use of nodes in public cloud. Otherwise encryption may be mandated to avoid data leaks.

5.3.4 Availability

It is expected that the nodes participating in a ledger offer guarantees for availability in a manner that matches or exceeds the requirements of the applications they serve. Certain applications may require a failover device ready for immediate activation.

5.3.5 Integrity and Security

The compute infrastructure may encounter malicious behavior, e.g. DDoS attacks. The infrastructure needs to continue guaranteeing the integrity of the compute service under these attacks.

5.3.6 Compute Device Characteristics

It is expected that the compute device meets or exceeds characteristics (CPU type, Operating System, Memory, etc.) required by the application and PDL type.

5.4 The Storage Vertical

5.4.1 Introduction to the Storage Vertical

Storage is the act of moving data through time. Storage may be local, remote, distributed or centralized, encrypted or not. PDL is distributed and decentralized by definition. Encryption is optional.

5.4.2 Per node storage

Though each node stores its own copy of the PDL data, the data storage itself does not necessarily need to be on the same node as the compute device. Data bay be stored on a directly connected storage device or on a NAS (Network Storage device).

5.4.3 Off-Chain storage

Depending on design and requirements, some chains contain all information always available, others allow off-chain storage of dated information an offer methods to retrieve off-chain information and verify its validity when required. Depending on the application and volumes of data processed, as well as the cost of storage and speed of access to off-chain data one may choose a storage option that fits its purpose.

5.4.4 Jurisdiction

Similar to physical location of compute nodes, depending on application and regulation the physical location of a storage device may be restricted to certain jurisdictions.

5.4.5 Privacy and Confidentiality

Certain entities (e.g. Banks) would only allow data to be stored within their own facilities and will not allow data storage in public cloud.

5.4.6 Availability

It is expected that Network Storage devices offer guarantees for availability in a manner that matches or exceeds the requirements of the applications they serve.

5.4.7 Integrity and Security

Network Data storage infrastructure is subject to both Device related and Network related risks. The infrastructure needs to mitigate such risks.

5.4.8 Storage Characteristics

Storage is characterized by volume, replication options, transfer rates and speed of access. It is expected that storage devices used in a PDL meet or exceed the requirements of the respective applications.

A typical application will use a mix of Compute, Storage and Connectivity functions to deliver its intended functionality. An example for such a mix would be a PDL application which requires storage, but also requires connectivity to convey data between the nodes as well as between storage facilities, and compute resources in order to manage user credentials, security, consumption, billing, etc. In certain scenarios the Platform Management Functionality may manage the resources in the Infrastructure layer.

6 Horizontal Integration

6.1 Definitions

This clause describes functions and concepts that are common across specific horizontal layers within the architecture defined in clause 4.

6.2 Similarities and differences between PDL and Legacy Databases/Ledgers

6.2.1 Introduction to comparative discussion of PDLs and Legacy Databases/Ledgers

This clause discusses the similarities and differences between a legacy, centralized, database/ledger and a PDL. Legacy ledgers are centrally managed though they may be distributed physically and geographically for diversity and resiliency purposes. Such legacy ledgers are owned by an entity that may be the same one using the ledger, or a third party operating a ledger used by another entity or entities. The key feature of a legacy ledger is its centricity: A single entity operating the ledger and responsible for the integrity of the information. A PDL, on the other hand, is not only physically and geographically distributed, it is also managed in a distributed manner. In most cases there is no single entity managing all nodes in a PDL though certain entities may each manage more than a single node in a PDL and in certain cases all PDL nodes are managed by a single entity. The integrity of the information in a PDL is managed through programing all nodes to use the same code, same software release, and same smart contracts. A consensus algorithm is expected to detect any node that uses an outdated software release or an outdated smart contract, or contains incorrect data, and eliminate use of this node until it is updated and aligned with the consenting majority.

6.2.2 Consensus Mechanisms

The most noticeable difference between DLTs is the Consensus Mechanism used to verify transactions and ensure the integrity of the chain. Consensus mechanisms vary in the algorithm used and in the rigorousness of tests required to verify a transaction. While in a permissionless distributed ledger the consensus mechanism should be very rigorous and should ensure there is a cost to fraudulent activity that is much higher than the possible gain from such activity, in a Permissioned distributed ledger the entities operating the nodes would typically have a common purpose and have a responsibility and an interest to maintain the integrity of the chain and a motivation to avoid malicious activity. Thus, the consensus mechanisms in such scenarios may use less rigorous methods to reach consensus. As an example, in a permissionless ledger a node has to "prove" it has done work, or has put funds at stake, that it will lose if it acts maliciously. In PDL nodes do not necessarily need to put anything at stake and may not be financially penalized if a record is compromised, as it is assumed that there were no malicious intentions. Consensus may then be reached through a reduced list of delegates where most nodes are simply transacting and delegate the consensus management to a select (voluntary or elected) list of validating nodes. It is beyond the scope of the present document to list all consensus mechanisms and algorithms currently in the market, and new algorithms keep being developed and introduced. Suffice to say that in a PDL there is typically a "double-safety" mechanism as there is the inherent tamper-resistance of the consensus mechanism and the mutual responsibility of the permitted validating nodes for the operation of the PDL, while in a permissionless distributed ledger the consensus mechanism is in fact the only means to prevent tampering with the information, and the nodes operate under mutual suspicion.

6.2.3 Basic Blockchain Operation

6.2.3.1 Discussion of basic Blockchain operations

The basic functions of a PDL are the following, most of which also apply to permisssionless DLTs.

6.2.3.2 Capture data transactions

The PDL captures a data transaction, or a group of transactions, or pieces of information, and packages them into a block. The minimal and maximal size of such block varies by PDL type. The data may or may not be encrypted. There is no restriction as to the source of such information or transactions, though the accurate capture of such information, sometimes referred to as "the last mile problem", may have significant influence on the quality and value of the PDL.

6.2.3.3 Hashing the block

Once a block is generated, a hash is generated from the sum of its contents. Hashes are created using the SHA256 or similar algorithms which are considered to be safe (taking into consideration that some hashes, such as SHA-1, have been compromised).

While it is possible (though with extremely low probability) that two completely different blocks will yield identical hashes, it is even less likely that a change of even only a few bits or bytes of data within a certain block will generate the same hash as was before such change.

As a result - once a block is hashed its integrity can be easily verified by repeating the hash (no matter if the information in the block is encrypted or not). If the hash produces the same result, the block can be considered untampered. If the hash produces a different result one should suspect that the block had been tampered with and may not contain the exact same information that was stored therein when it was originally created and hashed.

6.2.3.4 Chaining blocks

Blocks are chained to each other by including the hash of one block as part of the data of the next block. This creates a link and a dependency between consecutive blocks in a chain. If a block that is even a few links deep in the chain is tampered with, it will invalidate all subsequent links. Though it is theoretically (and practically) possible to keep altering the data in a tampered block in hope of finding a combination that yields the same hash, or otherwise re-hash all subsequent blocks so as to hide the tampered block, the computational resources and time required to perform such act without being noticed makes blockchains tamper-resistant. It is worth noting in that context that blockchain was considered "tamper-proof" in the past, but as it was proven that with sufficient time and resources it may still be tampered, it is now considered "tamper-resistant".

6.2.3.5 Reading data from blocks

The information stored in blocks can be accessed and read (unencrypting may be required) by entities that have access to the chain. In a PDL such access is restricted to permitted entities. Encrypted records within a block will require an encryption key that may be provided by the entity that has encrypted that record.

6.2.4 Other Blockchain Operations

6.2.4.1 Discussion of other Blockchain operations

In addition to the basic functions described above, PDLs offer additional functionalities.

6.2.4.2 Smart Contracts

Smart Contracts are computer programs stored in a DLT, wherein the outcome of any execution of the program is recorded on the DLT.

A smart contract may present terms in a contract in law and create a legally enforceable obligation under the legislation of an applicable jurisdiction.

Smart Contracts are discussed in depth in ETSI GR PDL 004 [i.10].

6.2.4.3 Zero Knowledge Proof (ZKP)

ZKP is the ability to prove something without having to reveal the information that such proof is based upon. ZKP may require extensive computational resources but is a useful tool in environments where competing entities use a shared resource.

6.2.4.4 Forks

6.2.4.4.1 Discussion of Forks

Forks are defined by ITU as "creation of two or more different versions of a distributed ledger" [i.5].

Forks occur when a blockchain diverges into two potential paths forward. Forks may be Intentional or Incidental, Hard or Soft [i.6].

There are numerous methods to handle forks (intentional and incidental) and it is beyond the scope of the present document to discuss this topic in much greater detail. Suffice to say that there are two possible solutions:

- One of the chains is continued and the other one is discarded, ending up with a single chain used by all nodes.
- The fork becomes permanent generating two separate chains, each used by a subset of the nodes.

6.2.4.4.2 Incidental Forks

Incidental Forks occur when two entities append a block (each appending a different block) to the chain creating two different chains in the same PDL. The result is that different nodes in the PDL may contain different chains. This is a situation that needs to be avoided.

6.2.4.4.3 Intentional Forks

Intentional Forks are the result of a change in the rules that govern the PDL. Such change of rules could be the result of a software upgrade or a change to a smart contract that governs the behavior of the chain. The result of an intentional fork would be similar to that of an incidental fork: different nodes in the PDL may contain different chains.

6.2.4.4.4 Hard Forks

Hard Forks are such that when a change to the protocol or rules that applies to one branch result in that branch becoming backward incompatible with the other branch of the chain.

6.2.4.4.5 Soft Forks

Soft Forks are such that when a change to the protocol or rules that applies to one branch result in that branch remaining backward compatible with the other branch of the chain.

6.2.5 Data storage and Privacy concerns

6.2.5.1 Discussion of Privacy concerns

Data Integrity and Privacy are a major concern in any platform used for storage and retrieval of data. The distributed nature of PDL, where data is duplicated in multiple nodes, poses an increased concern from multiple aspects.

6.2.5.2 Competition

A PDL may be used by competing entities which results in confidential information of one competitor being stored in a node maintained and owned by another competitor. The solution to such concern would be encryption of the information prior to creating the block, thus preventing unauthorized access. The inherent tamper resistance of PDL ensures an owner of a node cannot tamper information stored on their node. An example would be storage of confidential financial information of an entity in a PDL that has a node operated by a competitor of that entity.

6.2.5.3 Geography

Certain administrations regulate the geographical spread of information and may not allow nodes to be operated at certain locations. As an example, many countries do not allow personal identification information to be stored or even transported outside the borders of their country. As a result, a PDL that contains such information will be limited to the geographical borders of that country.

6.2.5.4 GDPR

GDPR and other Data Protection Regulations may regulate processing of personal information in ledgers. It is expected that PDLs that process such data comply with such regulations. Compliance may be achieved by different methods, depending on the specific regulation and type of information. An example would be removing the restricted information prior to creating a block and, possibly, generating a reference to where such information may be obtained in a manner that does not violate such regulations.

6.2.5.5 Storage in the Cloud and On-Premise

Regulations may state that certain information can be stored in the Cloud while other types of information are sometimes subject to more stringent requirements such as storage on-premise, in a private cloud environment, on a device owned and operated, bare metal, by the entity participating in the PDL. While it is not the intent of the present document to define or endorse such regulations, it is expected that node operators comply with such regulations and use private and public cloud only in scenarios where they are allowed.

6.2.6 Data Interfaces

6.2.6.1 Introduction to Data Interfaces

This clause will discuss the interfaces through which data is stored and retrieved from a PDL both on a platform level and a single node level.

Data loaded to a PDL by one node will propagate to all PDL nodes with time. The duration of propagation varies depending on factors such as geography (that affects network latency), consensus mechanism, number of nodes and others. For the purpose of the present document it is assumed that the information in all nodes in a properly functioning PDL is identical.

There are three primary data interfaces in a PDL.

6.2.6.2 Application to PDL

Applications access data in a PDL through an API to one of the nodes that operate the PDL. In the event that the application is running on the same device that is the PDL node - this is an internal API. In the event that the application is running on a different device the API will be using a communication protocol that both the PDL node and the application node are subscribed to. In most cases this will be the IP protocol over the Public Internet, but there may be scenarios where a PDL is implemented on an Intranet preventing access to and from the public internet.

In theory other communication methods can be used as long as all users and nodes in the PDL are using that method. Examples would be Metro-Ethernet, Frame-Relay and other OSI Layer-1 to Layer-3 protocols.

An application may use different methods to communicate with the end-user or edge device that generates or reads the data from the field. For example: A food quality application used by a consumer at a supermarket checking the freshness of food by scanning a barcode on its packaging using a mobile phone. The Barcode information is converted by the application into useful data that can be sent to the PDL for processing.

6.2.6.3 PDL node to Storage

A PDL node may use directly attached storage whereby the data interface between the PDL node and its information will be the device's internal bus or peripherals connectivity (e.g. Thunderbolt, SATA). In many cases the PDL node will be using a network storage device (NAS) in which case the information is accessed through an API or a specific network protocol between the PDL node and the data storage device. NAS can be implemented using any OSI Layer-1 to Layer-3 protocol.

6.2.6.4 Node to Node

Nodes would typically exchange information using the IP protocol over either the public internet or an intranet. However, Nodes may also exchange information through other protocols, as long as such protocols allow all nodes to communicate with each other.

6.2.7 Data Operation and Management

6.2.7.1 Introduction to Data Operation and Management

The distributed nature of a PDL poses challenges with management and operations of the data stored therein. A PDL allows selective access to its data, through membership management. A PDL also allows governance through a method agreed upon by the members. Such governance may be implemented through regulations imposed by a regulatory body or an external governing board, through a commission that governs software features or through development of specifications that software developers are required to meet when contributing code (either core or smart contracts) to a PDL. The consensus mechanism used by a PDL should reflect the governance implementation of choice.

6.2.7.2 Membership Management

Membership Management is achieved through rules that govern:

- How new members and nodes are added to a PDL.
- How and under what circumstances membership is revoked.
- The Anonymity/Pseudonymity of members.
- Access rights of members.

6.2.7.3 Governance

Governance is the method by which a PDL is developed and maintained. The functions being governed include:

- Software Development.
- Specifications/Standards/Regulations to follow such as KYC and AML.
- Authoritative teams such as a Board or a Software-Development-Advisory
- Choice of PDL technology.
- Choice of vendors.
- On-going operation and performance of the PDL.
- Membership Management.

A reference architecture for Governance of Standards is depicted in Figure 5 herewith.

Governance is discussed in further detail in clause 8 herewith.

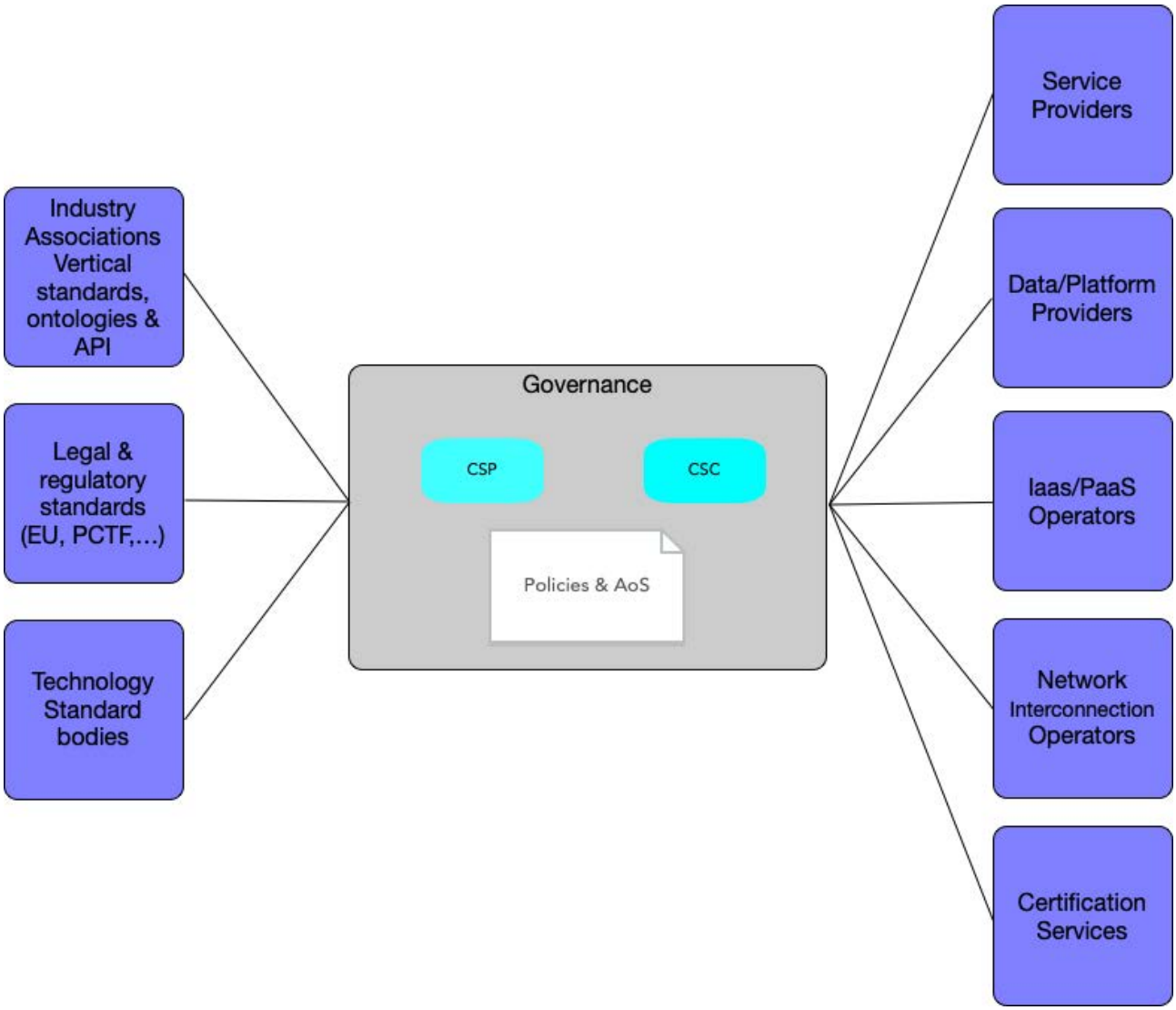

Figure 5: Architecture of Standards Governance

CSP, for the purpose of Figure 5, means Cloud Service Provider.

CSC, for the purpose of Figure 5, means Cloud Service Consumer.

AoS, for the purpose of Figure 5, means Authority of Surveillence.

6.2.8 Non-Permissioned/Permissionless Ledgers

Though the scope of the present document is Permissioned Distributed Ledgers, it is worth mentioning that many of the concepts and implementations of PDLs can also be applied to Permissionless ledgers. The reference architecture is one such example, where the abstract layers of a PDL are practically the same as those of a Permissionless ledgers and the differences lie in the implementations of membership management and consensus mechanism. Another difference, though not implicit, is the number of nodes, where a PDL will typically be limited to tens or hundreds, while a Permissionless Ledger may span thousands and more nodes. The performance would be the reverse of that where a Permissionless Ledger would typically be limited to roughly 10 TPS (Transactions per Second) while PDLs may be capable of processing thousands of TPS.

6.3 Platform Management

6.3.1 Inter-Node Communications

Inter-Node communications is typically handled using the IP protocol, which has intrinsic vulnerabilities. A node may potentially be spoofed and either send a message to, or receive a message from, a bogus/hacked node. There are several approaches to increase the trustworthiness of inter-node communications through methods such as:

- Encryption of the data using public and private keys.
- Using private networks (physical or virtual).
- Requiring the data to be received from multiple sources and comparing them all to ensure integrity.
- Suspecting all data received until verified through other means.

The last two point make up to what is otherwise known as a consensus mechanism.

6.3.2 Node Management

6.3.2.1 Node management consideratons

A PDL node is a computational device (dedicated node or virtual) that participates in a network of nodes through running an application (software). As such its management can be divided to hardware resource management, software resource management, storage resource management and governance management as described graphically in Figure 3 herewith. Policies and certification programs should be developed to describe the below requirements in detail. Such policies should be transparent and available to use by all stakeholders. The policies should be dynamic and development in a collaborative manner. The detailed policies, their development and maintenance, are beyond the scope of the present document.

6.3.2.2 Hardware resource management

Resource management for nodes implemented on dedicated devices includes:

- Power continuity - Ensuring power is continuous and UPS/Generation is available.
- Temperature management - Ensuring operational temperature remains within norms.
- Storage safety - Ensuring the environment does not pose safety hazards to the surroundings.
- Physical Access management - Maintaining access control to the device to authorized personnel.
- Diversity management - Ensuring diverse device types are available to overcome make/model vulnerabilities.
- Scalability and Extensibility Management - Ensuring devices can scale in number and resources with anticipated growth in demand or requirements.

When a node is implemented on a Virtual Machine Inter-VM it is expected that isolation is maintained on top of the above.

CPU resources and RAM, either dedicated or shared, are expected to be managed in a manner ensuring sufficient resources are available for the nodes' proper operation.

6.3.2.3 Software resource management

Software resource management includes:

- OS management: Ensuring the node is using a specified version of the Operating System and all required kernel extensions and security updates/patches.
- Application management: Ensuring the node is running a specified version of the PDL application. Upgrading the version if needed.
- Development management: Ensuring the development of software is carried out with efficiency:
  - Opensource - Interaction, co-development, access to code.
  - International Standards - Applying common adopted industry standards and methods.
- Security management - Ensuring proper risk assessment and mitigation model is applied.

6.3.2.4 Storage resource management

Storage resources are required for a node to store its local copy of the PDL data. Such resources are expected to be managed in a manner ensuring:

- Sufficient storage space is available for the expected volume of data.
- Data integrity is maintained (through e.g. replication, RAID, backups).
- Data is secured against unauthorized access and manipulation.
- Storage resources should comply with relevant integrity and confidentiality regulations and recommendations.
- In the event of network storage, not directly attached to the node, sufficient connectivity should be available to support the volumes and flows of data.

6.3.2.5 Node-Level Governance management

Governance management on a node level (there are PDL-level and Application-level governance aspects too, which are out of context go this clause) should define the following aspects as a minimum:

- Node Lifecycle: Depending on role of node (e.g. validator, regular) the addition or removal of a node to/from a PDL may vary both in process and in authority.
- Policies:
  - Resource policies: Define node-specific rules that provide probabilistic finality. (E.g. Always maintain sufficient resources for a specific functionality to be implementable. Raise flag is resources are unavailable).
  - Software policies: Maintain access control to the node.
  - Communication policies: Define rules that specify communication and messaging methods and process with other nodes under specific circumstances.
  - Connectivity policies: Define rules that specify connectivity methods with other nodes under specific circumstances.
- Standards: Where applicable standards should be adhered to for all aspect of governance.

6.3.3 Network resource management

6.3.3.1 Bandwidth Management

Bandwidth and throughput of segments of the network should be configured to be able to contain the load of traffic at peak utilization. Network upgrades should be planned and performed in advance to avoid congestion.

6.3.3.2 Performance Management

Network performance is managed through adherence to SLA performance metrics. Such metrics include attributes such Availability, Packet loss, Jitter and Latency. Different applications and stakeholders may have different SLA requirements and the Network should be capable of maintaining the SLA of the most stringent application (PDL or other) deployed thereon.

Resiliency, in the sense of failover/backup resources, may be required for certain applications.

6.3.3.3 Governance

Network Governance refer to the ability to monitor traffic on the network, allow or forbid certain traffic flows based on policy or regulations, ensure traffic is routed on permitted routes, ensure traffic is secured based on policies of regulations.

Network resource governance also refers to the ability to ensure the network is deployed using permitted network gear and technologies. Such permissions are based on policies and regulations and may vary depending on geography.

6.3.3.4 Inter-Node Network Management

Network resource management applies to the following scenarios:

- Ledger to storage: The network resources that connect a PDL node to its respective data. This could be an internal hard drive (implemented through the internal bus), a directly attached NAS (typically implemented using a dedicated copper or optical connector), or a remote NAS (typically using the public internet or a data-centre Ethernet fabric).

- Ledger to ledger: The network resources that connect PDL nodes to each other. This would typically by the public internet, but a PDL network can also be deployed over a private network using other technologies such as Ethernet, Layer-1 or others.

6.3.3.5 Inter-Node Network Management

Network resource security is discussed in clause 6.4.1 herewith.

6.3.4 Identity Management

Identity verification and management is an integral part of PDL, as well as many other aspects of our day to day lives. Compared to permissionless ledgers, where the identity of the participant is not known and none of the interest to other users, in a PDL, which is based on allowing only permitted users to transact (and thus possibly use less stringent consensus mechanisms) knowing certain facts about the identity of the participants and ensuring they are not using a false/fake identity is paramount to its proper function.

There are three scenarios pertaining to sharing of participant data between nodes and participants:

- All details: In this scenario all information is visible and shared with all other participants.

- Partial details: In this scenario certain information is shared with others and certain information is withheld according to applicable policies and regulations. E.g. When signing up on an on-line petition sharing of first and last name but withholding email address.

- "KYC-due-diligence-passed": In this scenario the participant will be identified by a key indicating it has passed KYC due-diligence (meaning its identity had been validated and is not fake/false) but the identity itself is withheld from other participants.

The choice of identity information to be shared depends on the application and the applicable policies and regulations.

There are multiple methods to ascertain an identity. The methods can be divided to Electronic (with near real-time verification) and Manual (which may require time to complete).

Electronic methods include:

- KYC processes vary by jurisdiction but serve as sufficient proof, according to that jurisdiction, that the identity of an entity (human, company, machine, etc.) is indeed what it claims to be.
- Digital records that can be attested as belonging to the person through biometric means (e.g. facial recognition compared to a photo-ID).
- Mobile-phone SMS return message.
- Code-Generator.
- Two-Step-Verification.

Manual methods include:

- Trusted-fellow referral.
- Company referral.
- Registration documents presented in-person.
- Postal verification.

Some of the above methods can apply as both Electronic and Manual.

6.3.5 Inter-Ledger Management

Inter-Ledger operations can be performed in at least two ways as depicted in Figure 6 herewith.

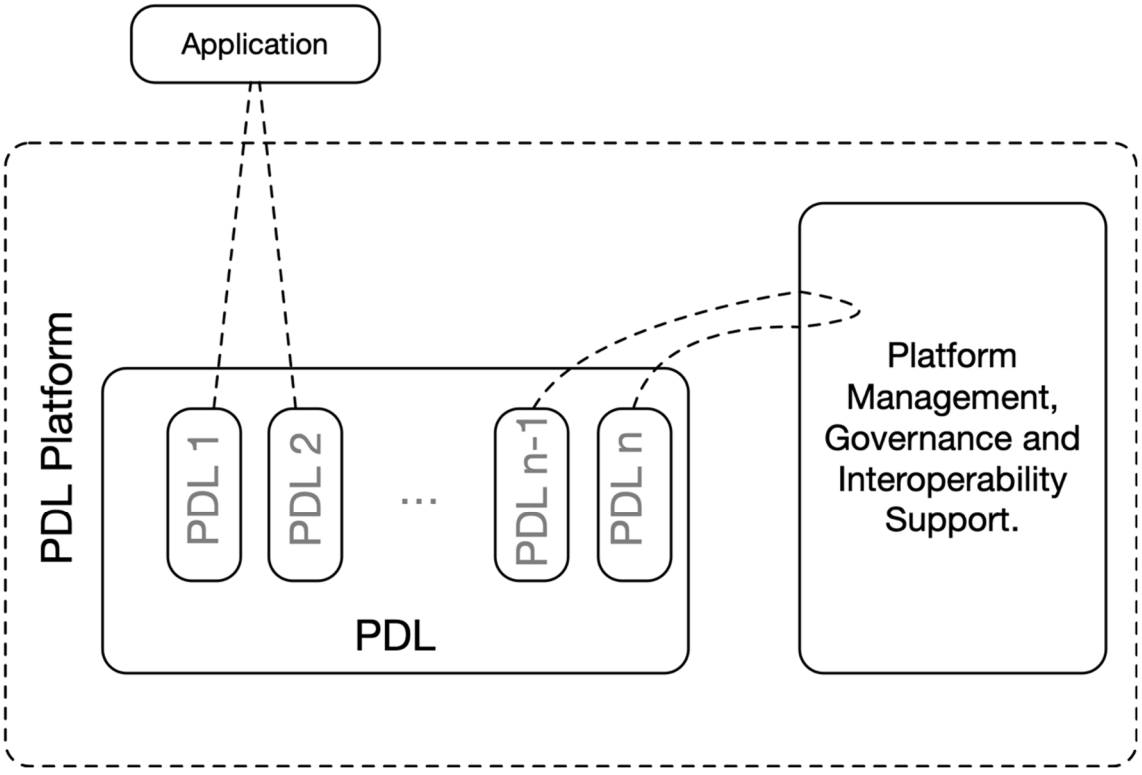

Figure 6: Inter-Ledger Management

1) Through an application at the Application layer. Similar to the "API Gateway" and "Oracle" approaches defined in the WEF document "A Framework for Blockchain Interoperability" [i.7].

2) Through the PDL Management functionality within the PDL Layer. This requires that the PDL Management Functionality is capable of managing multiple and different PDLs simultaneously. This is similar to the "Cross-Authentication" approach defined in that same document.

Inter-Ledger Interoperability is discussed in ETSI PDL 006 [i.11] and is out of scope of the present document.

Inter-Ledger Management can be performed through the governance functionality and implemented using smart contracts versioned for the different PDL types.

Information exchange across PDL types may be recorded on a third, neutral, PDL for census and validation.

6.3.6 Inter-Layer Management

Inter-Layer communications will typically be implemented through an API that serves as an abstraction between the two layers. Thus the Management of Inter-Layer functionality is highly dependent on the design and implementation of such API, the data-models applied on both layers towards such API and the process/program implemented on both sides of the API as a result of information exchange though such API. When both layere are developed in a coordinated manner the processes and data models will likely match by design. However, when each layer is developed independently, possibly by different and uncoordinated teams, there may be a need for a standard or and intermediator that ensures proper flow between layers.

6.4 General Considerations

6.4.1 Security

While security can be implemented as an application by itself (e.g. Firewall) this clause discusses the security aspects of applications in a more general term:

a) **Access-Control:** The ability to restrict access to or use of an application based on criteria such as identity, payment method and balance, location.

b) **Transaction-Security:** The ability to preserve confidentiality of information during execution of a transaction and the ability to prevent malicious or unintended execution of a transaction.

c) **Fraud-Prevention:** The ability to identify and mitigate fraudulent transactions.

Blockchain in general and PDL in particular are considered secure protocols. However, they are not hermetically sealed against fraud or malicious acts. The weak spots are divided into three families:

1) Consensus based attacks [i.8**]:**

a) **The 51 % Attack:** There is a risk that nodes that constitute a majority according to the consensus method will collaborate to bias records or record transaction priority breaching the integrity of the PDL. This is sometimes associated with bribery attacks, trying to buy votes in one's favor [i.9].

b) **Identity Theft:** If one's access information/details such as private key is acquired or stolen all the assets this person owns in the PDL will be compromised, and it can be nearly impossible to identify the thief. In PDL the information being compromised may include company records of multiple participants in the PDL, across entities. PDL may not be Tamper Proof as it was described in its early days, but rather Tamper Resistant and can be broken into given sufficient computational resources.

c) **Impersonation Fraud:** In absence of sufficient KYC or verification of identity procedures on a PDL, identity can be impersonated, exposing threats similar to those of Identity Theft or others that could affect the PDL and the Network's reputation. Such actions may be considered criminal depending on **jurisdiction.**

2) **System hacks:** While PDL data may be very difficult to hack and alter, the nodes and the code they execute is vulnerable to hacks and DDoS attacks. In addition to that PDL data that is transported through public internet is vulnerable to eavesdropping. System security (e.g. mitigation of DDoS attacks through KYC) should be applied to mitigate such threats. The detailed discussion of same is beyond the scope of the present document.

3) **Last Mile concerns:** Those typically related to collection and verification of data and it is recommended that those aspects are carefully designed and secured. E.g. an application associated with temperature control of a substance is dependent on the accuracy and functionality of the thermometers used.

4) Illegal activities:

a) It is possible that some participants of PDL issue transactions which are of illegal nature and in breach of certain laws. This problem is not specific to PDL, but the cryptographic nature of blockchain, specifically Permissionless environments, and its ability to store and exchange hashed information provides a fertile ground to same. Policies and preventive activities implemented in a PDL should identify and mitigate such activities.

b) Different jurisdictions may have different laws. PDL may be used to execute transactions that may be legal in one country and illegal in another. When PDL nodes span across multiple jurisdiction it is not straightforward to define under which law a PDL operates thus such actions may even be considered a-legal (subject to no law at all) and unregulated.

6.4.2 Economic Incentives

One of the key benefits of PDL is improved economic management. This is often evidenced in reduction of operational cost (HR, security, gear), Acceleration (shortening of identity verification and contract negotiation timelines, automation of processes), but in many cases it can also be realized through an increase in revenue from existing services/applications and from an ability to yield new revenues through the introduction of new services and applications that would be difficult to realize in absence of PDL technology.

6.4.3 Operational Incentives

PDL is often implemented as a replacement to complex platforms that may include multiple inter-dependent systems. PDLs provide accountability, all the transactions are recorded hence, can be verified later. The level and confidentiality details of information to be recorded depends on the consensus mechanism, security and other operational factors.

It has been demonstrated through several PoCs that a few lines of code embedded in a smart contract can replace the functionality of an entire systems. This adds to both cost savings, simplification and acceleration of existing processes.

6.4.4 Disintermediation

The industry as a whole and the wholesale supply chain in particular, could benefit from disintermediation. The Telecommunications industry, as an example, has a long history of operation in an equal-level playing field of federated operators. Telecom services such as international voice traffic and the public internet operate in absence of a centralized intermediating or controlling entity.

Distributed ledgers, both Permissioned and Permissionless, operate in a similar self-managed manner with no hierarchy of a top-level entity. There are certain wholesale supply-chain applications that rely on intermediaries (e.g. in the telecom world: mobile settlements) and PDL has the potential to replace such intermediaries by establishing trust between parties without the need for a trusted third party.

6.4.5 Identity Sovereignty

Identity Sovereignty can be self-declared or obtained through a governed KYC process. As described in clause 6.2.3 KYC processes serve as sufficient proof that the identity of an entity (human, company, machine, etc.) is indeed what it claims to be.

Self-Sovereignty is common in Permissionless environments but otherwise discourages or completely forbidden in permissioned environments such as PDL.

7 Integration Layers and Applications

7.1 Introduction to Integration

Any application would typically include at least one component of the three elements listed in clauses 5.1 to 5.3, and often an application will include a mix of all three. An example would be an application such as Dropbox™ that offers storage, but in order to provide such storage it also requires certain connectivity elements to move the data back and forth, and computational capabilities to identify users, calculate usage, perform billing and so forth. Another example would be a vehicle collision prevention application that uses computation to calculate its location and trajectory, and uses connectivity and storage to compare it with the location and trajectory of other vehicles in the vicinity.

Applications may be offered to individual users and enterprises, and can also be consumed or integrated by other stakeholders to generate other applications. As an example, a GPS positioning application may be used to indicate the geographical coordinates of an item. A mapping platform may be used to display a map of a certain place. Both but can be integrated into a navigation platform that shows the location of an item on a map, as well as propose a route from that location to another.

7.2 Templates

7.2.1 Template considerations

Templates may be used throughout the architectural stack defined in Figure 3 and may include one or more of the below items and functionalities. When designing a PDL one may use such templates to simplify design and re-use features and functionalities previously developed for similar purposes.

7.2.2 Applications

Applications may be re-used for multiple implementations. E.g. an application allowing SWIFT transactions through a bank can be used by multiple implementations that require such function.

Applications should be open sourced in order to provide transparency to the development community and the specific platform developer.

7.2.3 APIs

APIs and API gateways serve as a bridge between entities exchanging data and functionality. Their aim is to provide consistency across platforms and implementations. This includes a common, agreed-upon/standard-based structure and a model-based data format. API security compliance provides end to end security and prevention of fraudulent or malicious activity.

7.2.4 Common Functions

Common functions are reusable snippets of code, functions, smart contracts, libraries or even applications. Those functions should be available as templates for the benefit and re-use of developers and users of PDLs. Similar to applications, common functions should be open sourced.

7.2.5 PDL Implementations

There is no 'one-size-fits-all' for digital services. Multiple PDL types exist for a reason and the developer should have a choice to fit the purpose. A library of PDL templates allows choice and, in certain cases, also allows experimentation that leads to selection of the best fit.

7.2.6 Platform management and Governance

There may be different approaches to Platform Management in the form of governance, membership management, node privileges, fees, incentive plans, regulation conformance, consensus mechanism (to the extent possible within specific PDL implementations). Templates for same offer a menu of options for a developer to choose from and generate a best fit for purpose.

7.2.7 Infrastructure

Infrastructure recommendations should be available as templates that will allow the developer to ensure that the Computation, Storage and Connectivity infrastructure can support the software components and expected workloads.

7.2 PDL Middleware

7.2.1 PDL-Application abstraction

As illustrated in Figure 3 there may be, depending on the PDL implementation, a separation between the application and the PDL layer on which it operates.

In the event that the application is developed for a specific PDL implementation where the user or application interacts with the PDL layer directly it may be very difficult to abstract the application from the PDL. In the event that the application needs to be ported to a different PDL type, it has to be reprogramed almost in its entirety.

On the other hand, if the application operates independently from the PDL and interfaces with the PDL through an abstraction layer (e.g. API gateway) all southbound functionalities will use a unified method to access to the PDL, agnostic to the choice of underlying PDL type. Such abstraction layer could be implemented though an API-gateway other types of middleware. The benefit of such abstraction is that it allows separation of application from the underlying PDL so that an application developed by a certain vendor can be easily implemented on different PDL types.

Notwithstanding the above, PDL abstraction does not imply that all applications will or should be able to operate on all PDL types. For reasons of performance or features or regulation certain applications may not be able to operate on specific PDL types. E.g. A specific PDL type may not support the TPS rate (Transactions Per Second) that an application requires. E.g. The use a specific PDL type may be banned in a specific jurisdiction for regulatory concerns.

Applications should specify the features and attributes required from the PDL layer. This allows the platform developer to choose an appropriate PDL type that supports such features. The PDL abstraction layer may have the ability and logic to identify the available PDL types and use the appropriate southbound (PDL facing) APIs accordingly.

7.2.2 Application-Application abstraction

The functionality of certain applications can be implemented by multiple developers, each developing their own version of such functionality. In order for those applications to interface with one another they they are expected to be fully interoperable. Such interoperability may be obtained through a combination of three elements:

- Clear and unified application definitions. Each developer may add unique features that may prompt a customer to use their solution rather than the competitors' but the basic application functionality (often referred to as Minimum Viable Product (MVP)) will then be expected to be identical and fully interoperable.

- Unified and standardized interface reference points through which such applications exchange information with one another or other parts of the architecture. E.g. Eastbound interface between one application and another, Northbound interface between the application and the user.

- Unified information models that define the abstract processes and data models of the touch point between such processes at different entities across the interface points referenced above. When new products or services or applications (e.g. KYC) are introduced, the information model should be updated to include the required features so an enhanced standardized data model can be derived in order to enable exchange of the required information. The development of such standardized processes and Information Models is performed in Standard Bodies (e.g. ETSI, TM-Forum, ISO) or through industry collaborations that generate de-facto implementation agreements (e.g. Open Source initiatives).

Application developers are expected to abide by regulations and ensure the applications do not breach such regulations (e.g. GDPR). The application abstraction layer may thus be configured to restrict exchange of such information. Such regulations may vary from time to time and the Information Models, processes and abstraction layers should be modified accordingly to support such change.

An example that may illustrate this point would be vehicle tyres: when a driver arrives at a garage to replace a tyre there is a choice of tyres from multiple manufacturers, that meet the MVP (in this case the MVP is the circumference and width requirements of the tyre/wheel) and the consumer may choose the tyre that offers a price and quality (i.e. rubber grade) that meets certain budget or road-surface requirements.

The abstraction layer may also be used to implement security features such as access control, authentication, authorization or identity though such features may also be implemented on the application level.

7.2.3 PDL-PDL abstraction

Inter-PDL functions vary depending on the parity of the PDLs:

1) Both PDLs are of the same PDL type.

2) The PDLs are of different PDL types.

In the event of a pair of PDLs of the same type the inter-PDL functions are straight forward and may be implemented either directly on the PDL layer or indirectly through a management layer.

In the event of a pair of PDLs each of which of a different type, inter-PDL functions may need to be implemented through a management layer as it may be impossible to implement such management on the PDL layer itself.

The management layer interfaces with each PDL-type through an API. Such API, while likely being unique for each PDL type, will provide the full PDL functionality such as Governance, Access-Control & Security, Data read/write.

An additional aspect of PDL-PDL abstraction is ownership of the PDL Platform. When two PDLs are each operated by a different organization/entity/consortium they may be subject to different jurisdictions and regulations and thus may use a different vocabulary to read/write and store data. An API gateway may need to be deployed to make the necessary translations from one vocabulary to another.

7.3 Platform APIs

As illustrated in Figure 3, Platform APIs are internal to the PDL platform. They convey information between the functional elements of a PDL platform. Those include:

- APIs between the PDL management functions and the PDL (or PDLs).
- APIs between the PDL (or PDLs) and the underlying compute and storage platforms.
- APIs between the PDL management functions and the compute and storage platforms.

Some APIs may be considered partly-internal because they represent interactions with elements of the architecture that may, in certain cases, be internal, and in other external. Those include:

- APIs between the application and the PDL/PDLs (optionally, in absence of a PDL-Application abstraction layer).
- APIs between the PDL platform and the Templates. The templates may be integral to the platform management functionality or be external.

7.4 Service and Ecosystem APIs

Service and Ecosystem APIs allow a PDL to interface with external entities and functionalities such as other PDL nodes, application users (through e.g. GUI), other applications (e.g. digital wallet) and sometimes even the applications themselves (depending on availability of a PDL-Application abstraction layer). They can be grouped under the following:

- Inter-Application: APIs for the exchange of information between applications. E.g. An API used for money transfer between digital wallets.
- User-Application: APIs used to exchange information between the users of the application and the application. E.g. An API used to display the balance of a digital wallet.
- User-PDL: APIs used to exchange information between the user and the PDL will be implemented in the event that the application is embedded into the PDL. E.g. an API used to display the balance of tokens in a PDL.
- PDL-PDL: APIs used for exchange of information between PDLs. E.g. An API used for consensus between nodes.

Service and Ecosystem APIs can also be used for audit purposes in order to ensure the platform meets regulatory requirements.

7.5 PDL-based Retail Applications

PDL-based Retail applications are offered to end users, who may be employees of a company using the application to perform their day to day job or individuals using the application for their own personal use. Retail applications may be made available to end users through a wholesale supply chain or by a single supplier. The users would typically access and communicate with the application through a graphical user interface. The application will typically communicate with the PDL platform through the API and tooling abstraction layer though it may as well interface with the PDL platform directly. A PDL-based mobile wallet would be the immediate example of a retail application. On the backend such wallet will use the PDL to store value/balance of users' accounts and may also serve to identify users through KYC. On the front-end the wallet applications will allow the user to view their balance, initiate and approve transactions, as well as perform other user facing capabilities.

7.6 PDL-based Wholesale Applications

Wholesale applications are offered by and used by wholesale stakeholders as part of a supply-chain, forming an alliance of organizations/entities participating in the PDL platform. In a wholesale environment the participating entities may use the APIs and tooling layer to manage the PDL platform and run decentralized applications eliminating the need for a top-level orchestrator/intermediator. PDL serves as a trusted source of truth eliminating the need of a top-level "honest broker". PDL may also serve to exchange value between entities simplifying the complex bank-based fiat transactions, and eliminating the costs associated with SWIFT settlements.

In certain scenarios PDL may also be used to eliminate loss of goods, as it will immediately identify where in the supply chain goods recorded as delivered from one entity to another are not recorded as received by the other party.

A wholesale application may serve retail applications and may do so in a manner that is transparent to the retail customer. An example would be supply chain management of a mobile shopping web site, where the retail customer interface hides the wholesale supply chain through which the goods are offered and the payment route for such goods. Goods may be sold through a third party but appear to the consumer as though they are being sold locally. Additionally, the retail customer may be roaming internationally but may still enjoy a shopping experience as if they were shopping on their home network. On the wholesale side, the supply chain handles the resell of goods between vendors and the exchange and distribution of funds received from the retail customer. A wholesale application should also handle commercial aspects of supply-chain management such as mark-up, invoicing and settlement.

One example would be Decentralized Trustworthy Network Infrastructure (Recommendation ITU-T SG13 [i.3] Q2, see ETSI GR PDL 001 [i.4]), where Internet Service Providers (ISPs) participate the ledger to manage trustworthy Internet resource (including IP address, AS number, and domain name) ownership, and resource mapping (e.g. IP-to-AS mapping for BGP security, and domain-name-to-IP mapping for DNS security). This consistent trustworthy data on ledger is then distributed to other components in the network. In the BGP case, the IP-to-AS mapping data is directly downloaded to BGP routers, which is not aware of the ledger. In the DNS case, since the DNS client may not fully trust the DNS resolver, the client may also participate the ledger (as a lightweight verification client) to verify the resolution result returned by the resolver.

7.7 Interactions between PDL-based Wholesale and Retail applications

While inter-application interoperability is discussed in detail in clause 7.2.2, this clause specifically discusses interactions between Retain and Wholesale PDL-based applications. Such interaction does not necessitate that both the retail and wholesale applications use the same PDL, nor does it require that they both use the same PDL-type. It does require, though, that the unit of value being transacted by the retail application matches the unit of value recorded/transacted on the associated wholesale application. E.g.: When a consumer is purchasing 1 kilogram of tomatoes in a grocery store using a PDL-based wallet, the grocer's wholesale supply chain management PDL platform should record that 1 kilogram of tomatoes was sold and should be able to track the route of those tomatoes from the field to the consumer. Such an arrangement may be used, for example, to offer the consumer proof of origin of the tomatoes (e.g. bio, or date of picking).

The interface between PDL-based Wholesale and Retail applications had not been formalized nor standardized at the point in time the present document is written. It is thus open, at this time, for individual proprietary interpretations by the developers of the respective applications.

8 Permissioned Distributed Ledger Governance

8.1 The need for Governance

Permissionless ledgers, such as Bitcoin, are governed through the open-source code that runs them. PDLs, on the other hand, may use less stringent verification of transactions, and distribute tokens to participants without need to spend resources for PoW or PoS. Governance is then required in order to manage and operate the PDL. The functions governance should perform throughout the lifecycle of a PDL would be:

a) Selectively admitting new participants into the platform and granting read/write privileges.

b) Managing software development and deployment terms and conditions.

c) Establishing guidance, rules, policies and processes.

A PDL goes through three major lifecycle phases:

Creation of the genesis block and initialization of the PDL: This is a static phase that happens once. It includes the inception of the governing rules that would then apply to all PDL stake holders in the respective environment. Such rules can be embedded into the code that runs the PDL or may be added as a smart contract.

Evolvement phase: This is a dynamic phase through which the blocks are appended to the PDL under the consensus mechanism in convention, the PDL (optionally) forks to sub-chains or sidechains, smart contracts are (optionally) added and other. In the event that the consensus mechanism needs to be changed to a new one or other enhancement measures need to take place based on governance requirements, such changes will be agreed upon through consensus.

Termination: This is a static phase where the PDL is decomposed. In certain cases, Termination may require specific actions to be taken, such as surrendering all records to a certain jurisdiction or other actions in compliance with respective regulations.

8.2 Governance Methods and Structure

Governance covers four contexts:

- **Data:** Governance of the information stored in the PDL that is aligned and adapted to the lifecycle stages of the PDL in compliance with privacy and other regulations.
- **Protocol:** Governance defines the rules and protocols governing the behavior and alterations of the PDL during the lifecycle of the PDL.
- **Application:** Governance defines and ensures compliance with the rules for changes, maintenance and how different applications operate and interact with the PDL and guides how applications are terminated.
- **Institutional:** The governance defines how the PDL co-exists and interoperates with the organizational functions related to decision rights, accountability and incentives.

As stated in clause 8.1, Permissionless ledgers would typically be managed through development of code, which could be entirely open-sourced or guided by an entity.

PDLs can be governed either Off-Chain or On-Chain.

Off-Chain governance requires a governing body that could be:

a) A select group of representatives from the PDL membership (this select group may also include *all* members of the PDL).

b) An external panel/board of experts.

c) A mix of both.

The governing body may be appointed or elected or may consist of a mix of appointed and elected members. Election of board members can be based on seniority (e.g. Senior members' vote weights more) or size (e.g. Weight of vote is proportional to turnover or headcount of the voting member), or could be based on a single and equal vote per member. Other methods exist and are beyond the scope of the present document.

On-Chain governance differs in that IT systems allow to perform certain characteristics and dependencies of the PDL governance On-Chain. On-Chain governance requires a different approach that governs the adoption of automated functions and associated processes according to the degree of de-centralization. In particular there are dimensions such as decision rights, accountability, incentives, which are perfectly performed on-chain with the properties of immutability, transparency and administration of the PDL.

The governing body performs the governing roles defined in clause 8.3.

8.3 Governing the Governance

Governance involves the delicate task of both generating the rules and enforcing them. When bootstrapping a PDL based platform it is recommended that measures are taken to prevent a hostile takeover of the governing body in a manner that can lead to irreversible consequences.

Governance, in blockchain and DLT, is in principle affected by three factors:

a) **Governing Roles:** Roles that maintain security, robustness, availability, efficiency, continuity and accountability. Management of Governing Roles could be implemented through automated processes or by an external governing body defining the operational principles of the PDL.

b) **Target Audience and Developer Accountability:** A DLT can be Public or Private, each of which may separately apply to write privileges (users who may add records to a PDL) and read privileges (users who may only read records already on the PDL). Target audience is indirectly governed by targeting a specific audience, and methods can be applied to exclude certain (unwanted) audiences. Developer Accountability can be governed through unified incident mechanism and auditability principles.

c) **Permission Structure:** Governance can be applied through choice of permission structure between Permissionless or Permissioned. PDL is, by definition, a Permissioned environment.

Hybrid architectures are out of scope of the present document, but there are also governance mechanisms with hybrid consensus or multi-protocol environments which in essence get principles from these three fundamentals for a particular purpose generally.

History

Document history		
V.1.1.1	December 2020	Publication