INTERNATIONAL STANDARD

ISO 23257

First edition
2022-02

Blockchain and distributed ledger technologies — Reference architecture

Technologies des chaînes de blocs et technologies de registre distribué — Architecture de référence

Reference number
ISO 23257:2022(E)

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of ISO documents should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see www.iso.org/iso/foreword.html.

This document was prepared by Technical Committee ISO/TC 307, *Blockchain and distributed ledger technologies*.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

Introduction

Records of transactions, based on certain agreed upon conditions, form the basis for exchanging assets between parties. Businesses and governments have been operating for centuries using this foundation. While physical ledgers were once used, they have largely been replaced with modern technology. However, in traditional approaches, a ledger must be centrally controlled by one or a small number of parties, and other stakeholders must rely on them as agents to change those ledgers.

An important property of a ledger is verifiability. This means that the parties can verify that the set of transactions in the ledger is complete and accurate. As a result, these parties can identify irregularities in transactions, for example, to verify that digital assets of the participants are correctly accounted within a financial ledger. Currently, it is possible to achieve a verifiable ledger in a centralized way by making certain trust assumptions. However, verifiability can be also achieved by distributing the storage and decentralizing the control of the ledger with minimal trust in any one party.

By maintaining a ledger in a distributed network, Distributed Ledger Technology (DLT) systems, including blockchain systems, allow a much wider range of parties to have a shared view of the ledger and to make their own changes to that ledger.

A broad spectrum of DLT based business solutions is possible. This document presents a reference architecture for such DLT based solutions. It starts with the definitions and concepts of blockchain and DLT such as the system organization, nature of access, type of consensus and the roles and responsibilities of the participants. Given that the reference architecture must accommodate a wide variety of possible use cases, it touches upon various business domains and their respective use cases at a high level. Historically, ledgers have facilitated the exchange of assets, but DLT solutions can also be used more broadly for reporting, auditing, and coordination. The document finally presents the reader with various layers of a reference architecture for DLT systems and the functional components in the layers.

This document is relevant to, among other, academics, architects, customers, users, developers, regulators, auditors, and standards development organizations.

Blockchain and distributed ledger technologies — Reference architecture

1 Scope

This document specifies a reference architecture for Distributed Ledger Technology (DLT) systems including blockchain systems. The reference architecture addresses concepts, cross-cutting aspects, architectural considerations, and architecture views, including functional components, roles, activities, and their relationships for blockchain and DLT.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 22739, *Blockchain and distributed ledger technologies — Vocabulary*

ISO/IEC 24760-1, *IT Security and Privacy — A framework for identity management — Part 1: Terminology and concepts*

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO 22739, ISO/IEC 24760-1 and the following apply.

ISO and IEC maintain terminology databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at https://www.electropedia.org/

3.1
activity
specified pursuit or set of tasks

[SOURCE: ISO/IEC 17789:2014, 3.2.1]

3.2
architecture
fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution

[SOURCE: ISO/IEC/IEEE 42010:2011, 3.2]

3.3
behavioural interoperability
interoperability so that the actual result of the exchange achieves the expected outcome

[SOURCE: ISO/IEC 19941:2017, 3.1.6]

3.4
data archiving
digital preservation process that is moving data into a managed form of storage for long-term retention

[SOURCE: ISO 5127:2017, 3.1.11.19]

3.5
data flow
sequence in which data transfer, use, and transformation are performed during the execution of a computer program

[SOURCE: ISO/IEC/IEEE 24765:2017, 3.1006]

3.6
disruption
incident, whether anticipated (e.g. hurricane) or unanticipated (e.g. power failure/outage, earthquake, or attack on information and communication technology systems/infrastructure) which disrupts the normal course of operations at an organization's location

[SOURCE: ISO/IEC 27031:2011, 3.6]

3.7
distributed ledger technology governance
DLT governance
system for directing and controlling a distributed ledger technology system including the distribution of on-ledger and off-ledger decision rights, incentives, responsibilities and accountabilities

3.8
finality
property of a ledger that guarantees transactions in confirmed ledger records are irreversible and cannot be altered or deleted

3.9
functional component
functional building block needed to engage in an activity, backed by an implementation

[SOURCE: ISO/IEC 17789:2014, 3.2.3]

3.10
fungible
capable of mutual substitution among individual units

Note 1 to entry: The individual units can be digital assets, e.g. tokens.

3.11
governance
system of directing and controlling

[SOURCE: ISO/IEC 38500:2015, 2.8]

3.12
incident
anomalous or unexpected event, set of events, condition, or situation at any time during the life cycle of a project, product, service, or system

[SOURCE: ISO/IEC/IEEE 24748-1:2018, 3.22]

3.13
interoperability
ability of two or more systems or applications to exchange information and to mutually use the information that has been exchanged

[SOURCE: ISO/IEC 17788:2014, 3.1.5]

3.14
party
natural person or legal person, whether or not incorporated, or a group of either

[SOURCE: ISO/IEC 17789:2014, 7.2.3]

3.15
personally identifiable information
PII
information that (a) can be used to establish a link between the information and the natural person to whom such information relates, or (b) is or can be directly or indirectly linked to a natural person

Note 1 to entry: The "natural person" in the definition is the PII principal. To determine whether a PII principal is identifiable, account should be taken of all the means which can reasonably be used by the privacy stakeholder holding the data, or by any other party, to establish the link between the set of PII and the natural person.

[SOURCE: ISO/IEC 29100:2011/Amd 1:2018, 2.9]

3.16
policy interoperability
interoperability while complying with the legal, organizational and policy frameworks applicable to the participating systems

[SOURCE: ISO/IEC 19941:2017, 3.1.7]

3.17
provenance
information that documents the origin or source of an asset, any changes that have taken place since it was originated, and who has had custody of it since it was originated

[SOURCE: ISO/IEC 27050-1:2019, 3.19, modified — "Electronically Stored Information" has been replaced with "asset".]

3.18
resilience
capability of a system to maintain its functions and structure in the face of internal and external change, and to degrade gracefully when this is necessary

[SOURCE: ISO 37101:2016, 3.33, modified — Definition replaced with text in Note 3 to entry, Notes to entry deleted.]

3.19
role
set of activities that serve a common purpose

[SOURCE: ISO/IEC 17789:2014, 3.2.7]

3.20
semantic data interoperability
interoperability so that the meaning of the data model within the context of a subject area is understood by the participating systems

[SOURCE: ISO/IEC 19941:2017, 3.1.5]

3.21
sub-role
subset of the activities of a given role

[SOURCE: ISO/IEC 17789:2014, 3.2.9]

3.22
syntactic interoperability
interoperability such that the formats of the exchanged information can be understood by the participating systems

[SOURCE: ISO/IEC 19941:2017, 3.1.4]

3.23
transport interoperability
interoperability where information exchange uses an established communication infrastructure between the participating systems

[SOURCE: ISO/IEC 19941:2017, 3.1.3]

3.24
smart contract
computer program stored in a DLT system wherein the outcome of any execution of the program is recorded on the distributed ledger

Note 1 to entry: A smart contract can represent terms in a contract in law and create a legally enforceable obligation under the legislation of an applicable jurisdiction.

[SOURCE: ISO 22739:2020, 3.72]

3.25
consensus
agreement among DLT nodes that 1) a transaction is validated and 2) the distributed ledger contains a consistent set and ordering of validated transactions

Note 1 to entry: Consensus does not necessarily mean that all DLT nodes agree.

Note 2 to entry: The details regarding consensus differ among DLT designs and this is a distinguishing characteristic between one design and another.

[SOURCE: ISO 22739:2020, 3.11]

4 Symbols and abbreviated terms

AMQP	Advanced Message Queuing Protocol
API	Application Programming Interface
CAdES	Cryptographic Message Syntax (CMS) Advanced Electronic Signature
DLT	Distributed Ledger Technology
DNS	Domain Name System
EDI	Electronic Data Interchange
FBA	Federated Byzantine Agreement
GDPR	General Data Protection Regulation
HTTP	Hyper Text Transfer Protocol

HTTPS	Hypertext Transfer Protocol Secure over Socket Layer
ICT	Information and Communication Technology
IDE	Interactive Development Environment
IoT	Internet of things
IPFS	InterPlanetary File System
JSON	JavaScript Object Notation
MQTT	Message Queuing Telemetry Transport
P2P	Peer-to-peer
PBFT	Practical Byzantine Fault Tolerance
PKI	Public Key Infrastructure
RA	Reference Architecture
XML	eXtensible Markup Language

5 Concepts

5.1 DLT and blockchain systems

5.1.1 General

To understand blockchain and DLT systems, it is necessary to provide a description of the essential concepts associated with these systems and the range of distributed ledger technologies that exist.

A ledger is a long-established concept used in business and technology. When applied to ICT systems, it is an information store that keeps "final and definitive" records of transactions. Ledgers were originally and principally applied to financial transactions and to accounting practices. However, ledgers can be used to record transactions of almost any type: for example, the movements and transfers of physical objects.

A highly desirable property of a ledger is tamper-resistance, i.e. that transaction records, once entered into the ledger, are difficult to alter by design, and they cannot be altered without the alteration being clearly evident on inspection, whether the alteration is deliberate or accidental, malicious or benign.

The word "tamper-resistant" is more appropriate than "tamper-proof" since it can be extremely hard to prevent all forms of tampering. Similarly, although immutability is a design goal of DLT systems, it cannot be absolutely guaranteed. The word "tamper-evident" can be applied to a system that has the desirable characteristic of enabling any unauthorized changes to be clearly visible.

A distributed ledger has its entries stored across a series of nodes in a network, rather than in a single location. For example, the different nodes in the network can be owned, and interacted with, by different parties where each party has its own instance of records of only the transactions that it is involved in. DLT systems are designed to implement distributed ledgers, which is a significant challenge due to the need to agree on and maintain the transaction records in the distributed ledger.

With DLT, consensus ensures that every replicated version of a transaction is the same across all the nodes where it is stored – and that its contents are generally agreed amongst the parties involved in the transaction. The set of records in the distributed ledger should be verifiable and auditable. One of the major goals of DLT is to provide non-repudiable online transaction records.

Looking at DLT in this way does not imply that every node in the network stores exactly the same set of transaction records (although that can be the case for some forms of distributed ledger). It also does not imply that every party that participates in the distributed ledger has access to all the transaction records (it is possible that parties do not have access to transaction records they are not involved in). The transaction records stored on a node in a DLT system can be a whole or partial set of the distributed ledger implemented by the DLT system.

5.1.2 Blockchain DLT and non-blockchain DLT

Blockchain systems are a subset of distributed ledger technologies in which the state of a distributed ledger is maintained by processing batches of transactions in cryptographically secured data structures known as blocks. A valid protocol should ensure that each block is cryptographically linked to an immediately previous block forming a unique sequence of blocks in time. The complete sequence of cryptographically associated blocks forms a globally accessible append-only data structure - the blockchain - that provides the canonical version of the global transaction history.

In order to ensure that the ledger update process results in a single ledger state for a given block, blockchain protocols include a consensus mechanism that provides a total ordering of all transactions within the block. The collective action of all nodes in the blockchain system functions as a timestamp server that validates pending transactions and updates the current ledger state by sequentially appending blocks to the blockchain.

To implement a distributed ledger, blockchain systems require a mechanism to distribute new blocks to all nodes, a mechanism to validate transactions, and a mechanism to ensure consistency of all the copies of the blockchain.

The word “blockchain” is commonly applied both to the data structure, and to the complete implementation of a distributed ledger that uses the blockchain data structure.

A blockchain data structure can be used to implement something that is not a distributed ledger; however, such uses of blockchain are not within the scope of this document. This includes a centralized implementation of a blockchain database.

Not all DLT systems are blockchain-based. Some DLT systems use different ledger data structures with different approaches to storing transaction records and maintaining integrity, which typically offer different characteristics (e.g. capability to support high transaction rates).

In some non-blockchain DLT systems, a transaction record is stored as a separate ledger entry rather than as a part of the content of a block. Also, the ledger structure can possibly not be a chain. For example, there are ledgers wherein the underlying structure is organized as a directed acyclic graph (DAG) with transactions represented by vertices. The use of a DAG for transactions can improve the time and cost required for transaction validation but increase the effort for synchronization.

5.2 Networking and Communications

Networking and communications are an essential part of DLT systems.

A distributed program, or distributed application is an application that runs on a distributed system.

A P2P distributed application architecture partitions tasks or workloads between peers. Peers are equally privileged, equally capable participants in the application. They form a P2P network of nodes.

A DLT network is a network of DLT nodes that make up a distributed ledger system. Usually, DLT nodes communicate via P2P networks.

The protocol chosen to communicate between DLT nodes depends on implementation options and considerations available.

5.3 DLT platform

A DLT platform is a set of processing, storage and communication entities which together provide the capabilities of the distributed ledger system on each DLT node.

A node in this case is a machine in a P2P network that runs software components that communicate to support the distributed ledger and which can store a replica of the ledger. The node can either be a physical machine or else some form of virtual execution environment such as one or more virtual machines or containers. Virtual environments can be used if the node is implemented using cloud computing, for example. See ISO/IEC TS 23167 for more information about virtual environments and containers.

A node storing a complete replica of the ledger is referred to as a full node. Some DLT systems have nodes which, by design, do not contain a complete replica of the ledger.

A blockchain platform is a DLT platform where the implementation technology is a blockchain.

The capabilities supported by a DLT platform can include

— secure runtime environments,

— smart contracts,

— ledger,

— transaction system,

— membership services,

— state management,

— consensus mechanism,

— event distribution,

— cryptographic services, and

— secure inter-node communications.

The DLT platform is described in more detail in 9.3.5.

5.4 DLT system interfaces

In general, interfaces can be used for communication by users, by administrators, between nodes in networks, to smart contracts, between smart contracts, between DLT systems, and to external non-DLT systems.

To understand the appropriate interfaces needed for interoperability, it is necessary to first understand which systems and/or applications are exchanging information and for what purpose.

Figure 2 illustrates four common kinds of interfaces used in DLT systems:

— Intersystem interfaces A – directly between two (or more) DLT systems;

— External interfaces B – between DLT systems and external non-DLT systems – like DLT oracles;

— User interfaces C – between user applications and DLT systems;

— Admin interfaces D – between admin applications and DLT systems.

Intersystem interfaces support communication between separate DLT systems.

External interfaces can provide a secure means to access capabilities outside the DLT system such as trusted data sources or functions. Such outside systems include off-ledger code, DLT oracles, non-DLT applications and off-ledger data. These are linked to a DLT node using external interfaces. (See 9.3.2 for an explanation of how DLT oracles work.)

A DLT system includes both the user applications providing end-user capabilities and the admin applications that provide capabilities for administration and management of the DLT system. These applications access the DLT system via the user interfaces and the admin interfaces of a node, respectively.

Smart contract interfaces can be needed between DLT services, so that smart contracts on one DLT system can interact with another DLT system. In addition, users and administrators might need to communicate with smart contracts.

Interfaces both impact and support interoperability – see 6.7 for interoperability.

5.5 Consensus

Consensus in the context of DLT systems, addresses the problem of agreeing on the content and the order of records in a widely distributed system where new records could be competing to be added by multiple nodes across this network.

DLT systems can operate in a potentially geographically dispersed network of nodes. Each node might be able to create a new record to be included into the ledger or receive information about a new record. But at that moment in time, that new record might not have been seen by all other nodes in the system. In some circumstances, a node might receive information about two different new records, both competing to be the most recent record. Consensus mechanisms figure out how all these independent nodes in the DLT system come to an agreement about the contents and order of these records.

There are different aspects of consensus, usually resolved over different timescales. First, there is the question of consensus of what transactions or sets thereof are valid. This issue is usually resolved during the creation of a DLT system, then is accepted by nodes when they join the network, and can be updated by governance mechanisms over the lifetime of the network. The initial and ongoing acceptance of the validation mechanisms is often established by kinds of informal social consensus (public blockchains or DLT systems) or by contractual means (private blockchains or DLT systems). Second, there is the question of which valid transactions should be included in the most recent ledger record (and by extension, all subsequent records).

Many different mechanisms can be used to achieve consensus about the inclusion and ordering of transactions. The choice of an appropriate mechanism can depend on the possible threat model or failure modes to be guarded against by the network of nodes. Some of these mechanisms are given below:

— Round Robin: Nodes from a small group take turns as the authority on the creation of new records, perhaps in a round-robin fashion.

— Byzantine Fault Tolerance or Byzantine Agreement: Conventional approaches within the distributed systems literature often consider the threat of “Byzantine failure”, where individual nodes in the network might not only be delayed in hearing new information from other nodes but might also be sent maliciously-constructed information from (a bounded number of) other malicious nodes. In this setting, a variety of “Byzantine fault-tolerant” consensus algorithms have been proposed, including Byzantine Paxos and PBFT. Normally, these require the number of nodes in the network to be known, for the number of malicious nodes to be small (typically, less than a third of the total number of nodes) and for every node to establish evidence from a quorum (typically, a clear majority) of other nodes. These kinds of mechanisms are often used in private DLT systems. In practice, good performance for these mechanisms might limit the size of the network to tens to hundreds of nodes.

— Nakamoto Consensus: For public blockchain systems, the participating nodes might be unknown, and their number can vary. Therefore, the nodes that should achieve consensus might be unknown. The consensus mechanism used in the Bitcoin blockchain, and in other public blockchains, such as

Ethereum, is known as “Nakamoto Consensus” and it addresses this challenge. The general scheme is that every node will accept as authoritative the longest sequence of blocks that it has seen. In the case where there are competing alternative blocks, there can be short-lived alternative histories (“forks”, or “uncle blocks”) temporarily accepted by various nodes within the network. However, it is likely that nodes eventually converge on acceptance of a common blockchain history, at least for older parts of the ledger. Some versions of consensus use not only the length, but also the total difficulty for Proof of Work, so the assessment can be multi-dimensional.

Proof of Work and Proof of Stake are two ways of mining a new block to accomplish Nakamoto Consensus:

— Proof of Work: The creation of new blocks in Bitcoin requires that a difficult cryptographic or computationally hard puzzle is solved: given a set of transactions for a new block, choosing a nonce that will make the hash-value for the block smaller than some currently-accepted difficulty target. Although finding a solution is hard, checking the solution is easy. The overall approach is called “Proof of Work”. There is no known efficient solution to this kind of problem, and so a brute-force approach needs to be used. This means that the time required to find a solution is essentially random but varies in proportion to the computing power available to solve the problem.

  Proof of Work requires the investment of significant monetary capital into the computational resources needed to solve the problem. The nodes could receive an immediate return by being awarded both newly-minted cryptocurrency that can be claimed as a block reward, and transaction fees offered by individual transactions included in the block. This investment and return tend to align the incentives of the node with the creation of value and integrity in the overall DLT network. This scheme can accommodate any number of nodes in the network, by iteratively adjusting the puzzle difficulty over time in response to changes in the average time required for the network to solve each puzzle.

— Proof of Stake: In this mechanism, random leader election is determined by a kind of bet made in proportion to the amount of cryptocurrency staked by nodes competing to define the next block. The stake holding of nodes in cryptocurrency serves to align the incentives of the nodes with correct functioning of the network.

Other approaches for consensus are possible in permissionless DLT systems. For example, all nodes might periodically elect a small known number (tens to hundreds) of nodes to act on behalf of the whole network. Then, conventional consensus mechanisms that are normally appropriate for permissioned DLT systems can be used by the elected nodes. Given this, a reward system or incentive mechanism is a consideration (such as a payment) given to a party for undertaking some activity within the DLT system. An example of such activity is the work involved in running the consensus mechanism. Mining is an activity to seek rewards in some consensus mechanisms, and in those systems, a miner owns a node that runs mining programs.

Finality is a property of confirmed transactions that indicates whether they will remain part of the ledger after confirmation. Some DLT systems can have multiple non-identical blockchain histories. As the DLT network converges on a common blockchain history, the other copies of the blockchain history are discarded and as a result, some confirmed transactions might no longer be part of the ledger. This is particularly evident in DLT systems that use Nakamoto Consensus as the nodes will discard their current blockchain history for one that is longer. Thus, a confirmed transaction does not necessarily end up on the ledger. As external systems often depend upon a confirmed transaction being part of the ledger, a common practice is to wait until a certain number of blocks have been added to the ledger before concluding that the transaction will be part of the ledger. However even after waiting some number of blocks, there is still a possibility that some longer blockchain history will be uncovered that doesn't include the transaction. This type of finality is referred to as probabilistic finality and is in contrast to immediate finality used by DLT systems that cannot have multiple blockchain histories for a single ledger. With immediate finality, external systems know that once a transaction has been confirmed, it is guaranteed to be part of the ledger.

5.6 Events

An event is a certain situation occurring at a certain point in time. In the context of information systems, an event can be understood as a change of state or value detected for processing and/or reporting (see ISO 16484-2:2004, 3.74). For example, it could be a message sent from a node to the external application. Events signal external stimuli, changes to field values, and interactions between nodes (see ISO/IEC 14772-2:2004, 3.3).

From an architectural point of view, one can distinguish

a. internal events - internal to a DLT system, and

b. external events - communicated to and/or recorded in a DLT system.

Like other IT systems, a DLT system should ensure proper handling, logging and traceability of internal events in order to ensure security, auditability and transparency. The functionality of the system should need to include event management functions supporting pre-defined or custom event services for DLT users. (see, e.g. ISO/IEC 17789:2014, 8.3.2.2, second list item). Recorded information about events should be tamper-evident, whether an event is recorded on-ledger or off-ledger. Where modifications are needed, that can be done by means of adding new records while preserving previous ones for the sake of accountability and transparency. Within the context of DLT, the ledger only guarantees security for on-ledger data. To guarantee security for off-ledger data, separate security measures should be in place.

External events are vital for smart contract event-driven functionality. Connection with the outside world is established by data exchange including the data about the events. It is essential to ensure the trustworthiness of the sources of external information (called DLT oracles). Trustworthiness of the external sources can involve non-technical considerations. Requirements for trustworthiness might be established, for example, by relevant legislation, by DLT system policies, or by agreement between the parties to a smart contract.

Since events are fired by publishers and received by subscribers that had previously registered to receive them, this form of communication is inherently asynchronous. Depending on the importance of the system on receiving the events in a reliable and/or timely manner, systems can be put into place to handle any network or node failures, i.e. buffering and forwarding events that are not received or acknowledged where appropriate. Many different event systems are reliable, well-understood, and appropriate for use for DLT systems and are therefore not further explained.

Examples of events to consider include the following:

— network infrastructure events: ledger network deployed, joined or health status updated;

— node events: ledger node deployed, updated, health status updated (online/offline);

— user events: DLT account created, refreshed, user login;

— smart contract events: smart contract deployed, smart contract instance created;

— transaction events: smart contract instance function executed, updated, smart contract instance state updated.

5.7 Integrity of ledger content

In general, integrity implies completeness and not having been altered or deleted in an unauthorized way. In DLT, there are many characteristics related to integrity, such as consistency, immutability, and accuracy. For DLT systems, integrity can be considered at various levels of architectural abstraction. At the level of an individual transaction, integrity means validity, in the sense that the transaction complies with the DLT system's rules enduring consistency for its inclusion in the ledger. At the level of the ledger, integrity usually means being complete, immutable and only created according to the DLT platform's rules. In this case, immutability means that only new transactions can be added to the ledger once consensus has been achieved and existing records cannot be changed. At the level of applications

using the ledger, integrity will be defined with application-specific criteria. These can even include qualities that are not capable of being mechanically enforceable or verifiable, such as being accurate.

One of the key goals of DLT is to provide secure transaction records that are the same for all participants in a given multi-stakeholder scenario, and that cannot be repudiated without requiring the use of a centralized database.

In some DLT solutions, provenance information can be recorded on the ledger using one or more related transactions. Provenance is a record of ownership, origin, and/or location of an asset. DLT systems are useful for applications that track provenance because all transactions relating to an asset are maintained in a tamper-evident ledger.

In some supply chain models, the data associated with the creation and movement of the assets, if tracked, are registered, and maintained separately by individual businesses involved in the supply chain. There is the potential for the information in these systems to differ and in addition there is no guarantee that the information maintained is immutable. This makes tracking an asset difficult, time-consuming and potentially unreliable. The inability to easily and reliably trace the origin and track the path of an asset has major implications for businesses. In the case of a food supply chain, a large amount of uncontaminated food might have to be discarded along with a small amount of contaminated food during food recall. Counterfeit drugs whose origin is unknown in a pharmaceutical supply chain could result in deaths.

As the assets are created and exchanged through the supply chain, transactions describing these events are recorded immutably into the distributed ledger, with each participant having access to a local copy of the transaction records. The distributed and tamper-evident goals that are fundamental to DLT systems (as described in 5.1) make them suitable for these purposes.

5.8 Integrity and ledger management

For integrity to be preserved and the DLT system to be trusted, once consensus is achieved on transaction records or content that are added to the distributed ledger, their immutability should be ensured. Individual records should not be deleted.

Deleting means permanently erasing and removing validated records from the distributed ledger. DLT systems preclude deletion by design.

Since DLT systems are append-only systems and transaction records are not removed, the distributed ledgers can get too large and unwieldy to manage them efficiently. Pruning and archiving are methodologies to place the oldest transaction records into some forms of archive storage and remove them from the live replicas of the distributed ledger. The archived or pruned records remain available on demand, although access to them could be considerably slower than for live ledger records.

Pruning a distributed ledger can be done by creating a smaller replica of the distributed ledger by filtering out transaction records meeting specified criteria. Those filtered transactions should be able to be restored with integrity if needed. Criteria for pruning could include being older than some specified time or having no more output to spend.

Pruning can be done for some subset of nodes in the DLT system. Full replicas of the ledger at other nodes would permit the retrieval of transactions that might have been pruned (see archive, backup, and restore) and would maintain the overall integrity of the system.

Given the distributed nature of the DLT systems and the implications of archiving and backup operations for trust and integrity, it is essential to have robust resilience and governance to include these operations to enable recovery in the case of disruption.

Data archiving can be done to improve resilience and trustworthiness of the system. Pruning does not preclude archiving, backup and recovery from being designed into the system as well. Data archiving is a digital preservation process that moves data into a managed form of storage for long term retention. An archive is set of data, including record entries, transactions or resources saved for later reference or use, possibly off-line.

A common method to preserve data, independent of pruning and archiving, is using backup and recovery of the ledger or node. In this case, backup is the process of copying/exporting transaction records or other data, to the data storage of a backup system to enable retrieval and restoration of these data or re-execution of the transactions in case of an incident or disruption. The copy is referred to as a backup copy. Restoring should be done in a way that preserves the integrity (same number and content of all records) and content of the distributed ledger.

5.9 Subchains and sidechains

A subchain is a logically separate chain that forms part of a DLT system. It is also known as a channel in the architectures of some DLT platforms. Each subchain could be owned by a different entity and could be accessible to a different set of users. Nodes could be set up so that some nodes participate in certain subchains and not in other subchains. The result of this configuration is that the ledgers on some nodes can contain transactions for a given subchain while the ledgers on other nodes do not.

A sidechain is a separate blockchain system that runs parallel to the original blockchain system that can participate in transactions with it typically in both directions, using a two-way interface. The two-way interface enables interchangeability of assets at a predetermined rate between the original blockchain system and the sidechain.

5.10 DLT Applications

A DLT application, also termed a ledger application, is an application that provides functionality supporting some activities of DLT users. A DLT application can be a user application for users or an admin application for administrators.

The DLT application interacts with a DLT system via the User API made available by a DLT node and operates using the DLT system and sets of capabilities provided by non-DLT systems needed to deliver the functionality of the DLT application. The User API can offer operations that can be invoked by the DLT application and is also an interface for the DLT application to receive events happening within the DLT system, for example, specific transactions being recorded on the distributed ledger. A DLT application can initiate a transaction within the DLT system. In some cases, the User API allows communication with one or more smart contracts where the implementation of individual User API operations is provided by a smart contract.

DLT applications can support users via typical user interfaces such as desktop, web, mobile, and console. DLT applications can also be automated. Examples of automated DLT applications include those which connect IoT devices to DLT systems, for example where the IoT devices are detecting events in the physical world and need to record these events in the distributed ledger – in these cases, there might be no human users of the DLT application under normal circumstances.

For admin applications, the DLT application is used to administer the DLT system itself, or to administer individual DLT nodes. These administration DLT applications usually interact with the DLT system via specialized administration APIs.

5.11 DLT solutions

DLT solution is the term applied to a complete end-to-end solution that is built using a DLT system to accomplish some objectives relating to a group of organizations. An example is a solution supporting supply chain trade facilitation, dealing with many different organizations involved in a supply chain, such as producers, customers, shipping organizations, government authorities and so on.

DLT systems are distributed by definition, with the DLT system having a number of DLT nodes networked together. DLT solutions are similarly distributed, with numerous components distributed over the network.

It is typically the case that the different DLT nodes of a DLT system are decentralized, meaning that they are controlled by different organizations (or individuals) and it is possible for each DLT node to be used by different DLT applications. Each DLT application has functionality that reflects the activities

and the concerns of the related organization and its users. It is of course also possible that the same DLT application is used by multiple organizations and their users, where they are all performing the same activities.

In addition, the capabilities of the DLT system itself could be defined by the smart contracts that exist on the DLT nodes, which are responsible for handling transaction requests and for adding transaction records into the distributed ledger. A DLT application invokes one or more smart contracts through the user API to initiate a transaction. A DLT solution can also have a set of DLT oracles and associated non-DLT systems.

Thus, the DLT solution consists of the DLT system with its DLT nodes and communication networks plus all the DLT applications connected to each of the DLT nodes, along with any associated non-DLT systems connected to the DLT system. The arrows represent data flows between a DLT system, DLT applications and non-DLT systems, as shown in Figure 1.

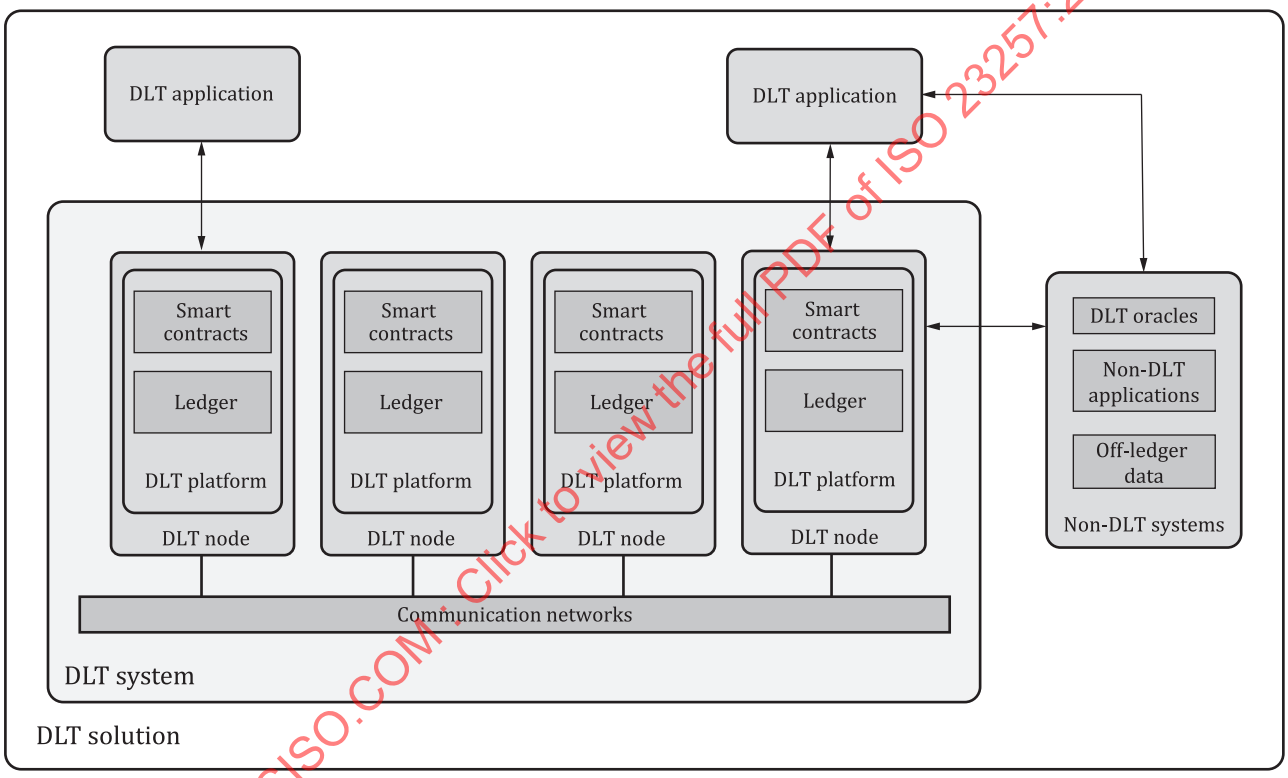


Figure 1 — DLT Solution and its constituent parts

5.12 Smart contracts

5.12.1 General

Not all DLT systems use smart contracts. An extended description of the implementation, execution process and life cycle is given in ISO/TR 23455, therefore at this point, only the architectural components will be described.

The logic associated with smart contracts should execute deterministically, otherwise no consensus about the result of a correct execution will be achieved.

Most execution environments allow the smart contract to be programmed in high-level or domain-specific programming languages. Thus, smart contracts can have variables to store data and functions which access, process, and write data.

In some systems, smart contracts may be executed on every node, and in other systems they may be executed on only some limited set of pre-specified nodes.

Additionally, since the execution of code by miners or validators can prevent them from working on their main tasks – creating and validating blocks – the number and timing of executing smart contracts has to be very limited to enable a fully functional DLT system. Furthermore, data-intense applications need to store and retrieve their data off-ledger to prevent a denial-of-service situation in the DLT system. Newer architectures are emerging that can handle both tasks in a way that does not impact the system.

5.12.2 Smart contract execution on dedicated peers

Smart contracts can be locally installed and executed on dedicated peers. Locally executed code can store or retrieve all processing requirements for the execution with a deterministic outcome on the local peer.

This local code leaves a security risk in terms of a singular point-of-attack as it can be altered before or even during execution by a malicious actor. Therefore, an additional validation stage has to be executed by another trusted instance to verify the validity of the results such as computed by the above-mentioned peers. In this kind of execution environment, blockchains and DLT systems can directly interact with their peers' environments via device drivers as the point of execution is known.

5.12.3 Smart contract execution on arbitrary peers

Code executed in an arbitrary location reads the state from the ledger as this is the only consensually agreed location for deterministic status quo. As this state and all changes to it again replicate throughout the whole P2P network, memory consuming transactions should be avoided in smart contract systems running on arbitrary peers.

Smart contracts that execute on arbitrary peers require three to four potentially different roles.

— The smart contract needs to be recorded to the DLT system by a so-called smart contract owner. This owner has extended rights to the operation of the smart contract such as deactivating it permanently, also called "destroying" it;

— A potentially different, second party might now invoke the code by sending a transaction to its address;

— A third party, known as a miner or validator in some DLT systems, instantiates and executes the code and records the result on the ledger;

— This result shall be checked by other miners or validators while validating and appending a block containing the execution result. For security reasons, computing the outcome and checking it might be done by different miners or validators.

The smart contract code itself can be deployed via a transaction to the distributed ledger network, where it undergoes the usual validation process such as mining and consensus. Once being added to the distributed ledger, it is practically immutable; however, some DLT systems provide features that allow the use of an updated version of the smart contract.

As the location of execution is arbitrary, smart contracts can interact with IoT devices, adapters, or third parties via external interfaces, usually called DLT oracles, addressable on dedicated peers of the DLT systems.

5.13 Transactions and how they work

For the purposes of this subsection, a transaction is understood as an atomic (indivisible) change of state recorded in a distributed ledger. Integrity requires an invariant condition for the whole DLT system to be maintained while executing transactions. A transaction changes the system from one

consistent state to the other. Atomicity of a transaction means that it will either succeed or fail but will not be only partially completed.

For public DLT systems, the simplest kinds of transactions are the transfer of cryptocurrency or other digital asset from one address to another address. In such systems, the state of the DLT will record the control of assets by addresses. Important rules for the transfer of digital assets typically include that the original address controls the assets being transferred. The rules for some DLT systems only allow transfers from a single source address to a single destination address, while other systems allow transfers to and from many addresses in a single transaction.

On some DLT systems, some transactions can have more impact. For example, on Ethereum, transactions can record the results of executing smart contracts. For Ethereum, the rules include a specification of the execution of Ethereum Virtual Machine instructions that constitute the object code for those smart contracts.

On some DLT systems, a transaction's change of state can include recording information or events.

The results of executing transactions can be affected by the state changes due to other transactions. So, the order in which transactions are executed is important. For this reason, transactions are performed sequentially, or at least can be viewed as having been performed sequentially. However, sophisticated transaction evaluation mechanisms can allow concurrent evaluation of transactions while maintaining serializability. This can be done if there are no interdependencies between transactions. However, determining that two transactions have no interdependencies can be difficult, especially for transactions which execute smart contracts.

Some transactions may only be executed by individuals in certain roles within a contract or holding a specifically authorized on-ledger identifier, e.g. an address in the DLT system.

5.14 Tokens, virtual and cryptocurrencies, coins, and associated concepts

DLT systems sometimes, but not always, focus on a unique tracking mechanism, broadly in the category of tokens. Historically, in non-DLT systems, physical tokens (such as coins, chips or tickets) have long been used within local closed systems or environments. Currency and coins can have intrinsic value (which can change value dramatically). Tokens like chips, tickets, stamps, armbands, and other devices are issued for local uses, such as at a movie theatre, for a subway system, within a music festival, for arcade games, gambling casino, or to use washers and dryers at a laundry - to simplify handling of cash within the system. They can also minimize the risks of monetary handling, and, for psychological purposes, reduce inhibitions to spending, reduce risks of tokens being stolen to be used elsewhere, and raise the barrier of converting tokens back into cash.

Cryptographic tokens, virtual and physical, have likewise long been used in various distributed system environments, primarily for access to a system; they have taken on numerous new roles in a DLT environment, where the rights of the holder are not overseen centrally.

Assets are items of value that can be physical and virtual. Digital assets are assets that only exist virtually or which are the digital representation of other assets. Cryptocurrency is associated with building value for virtual coins or tokens.

A token is a digital asset that represents a collection of entitlements. Generally, asset backed tokens are registered in ledgers and are then associated with users.

Categories of tokens include the following:

— Intrinsic value tokens: Where the value of the token is based purely on the supply and demand for the token itself, where it becomes a "native currency" within some environment;

— Extrinsic value tokens: Where the value of the token is based on some external reference point. Like the physical tokens, the token is a proxy for something else: another token, a physical asset or virtual assets;

— Representative (non-value) tokens: Tokens can be designed for many other purposes within a system, just as they did in the physical world, where the primary goal is not limited to exchange of value. Tokens might be used as tracking devices.

Electronic coins are primarily designed for payment systems (e.g. Bitcoin). Electronic coins are primarily designed for payment systems (e.g. Bitcoin).

Cryptographic tokens, virtual and physical, have likewise long been used in various distributed system environments, primarily for access to a system; they have taken on numerous new roles in a DLT environment, where the rights of the holder are not overseen centrally. For example, tokens can be used to track usage or be used as incentive tools for higher quality and governance enforcement, or in some cases they can represent other rights such as beneficial interest in a financial instrument.

“Virtual Currency Schemes" published by the European Central Bank (ECB), in 2012, defines a virtual currency as a type of unregulated, digital money, which is issued and usually controlled by its developers, and used and accepted among the members of a specific virtual community. See Annex A for a full discussion of tokens and currencies.

6 Cross-cutting aspects

6.1 General

Cross-cutting aspects are behaviours or capabilities that are coordinated across roles and should be implemented consistently in a DLT system. Cross-cutting aspects can be shared with and can impact multiple roles, activities and functional components.

Cross-cutting aspects of DLT include

— security,

— identity,

— privacy,

— governance,

— management,

— interoperability, and

— data flow.

6.2 Security

Since DLT systems heavily rely on distributed communication, distributed nodes, consensus mechanisms and cryptography, the following security elements need to be taken into consideration:

a) network security;

b) choice and configuration of cryptographic algorithms and protocols;

c) cryptographic key management;

d) application security;

e) security management processes;

f) secure implementation and certification;

g) availability.

DLT specific security considerations at issue include

a) consensus security,

b) hard/soft fork management, and

c) block data management.

Membership services manage the membership of DLT systems in consideration of identity, privacy, confidentiality and auditability within a DLT system.

Role based access control is sometimes used to provide a more expressive model for access control policies systems.

As the ledger grows, key management could be challenging. For example, if a private key is compromised, changed or revoked, then it will be hard to trust and process data previously signed with this key. Sometimes, when a user ID is based on the user's public key, this could lead to challenges with the user identification. In order to override this, specific key changing techniques need to be implemented. For private DLT systems, the mentioned problems could be solved by incorporating PKI, CAdES, etc.

In a DLT system, information security includes the protection of the confidentiality, integrity, and availability of information, and the protection of PII, and it covers more than the ledger. It is a cross-cutting aspect and covers the user, non-DLT systems, API, DLT Platform, infrastructure and other cross-layer functions. Understanding the security considerations helps to select and integrate security mechanisms to protect the hardware, middleware and software that makes up the DLT system. Cross-cutting security functions include asset protection, access management, identity management, cryptography management, assessment and testing management, PII protection, and availability management.

Each security function can be composed of one or more services, processes and tools, which can be deployed to meet organization requirements across the layers and functions. For example, access control services can be deployed in the user and non-DLT systems layers, in the development functions, and in the management and operation functions.

Organizations can deploy or implement all, some or none of the components of the architecture. The selection of architecture components, services, processes and tools adopted can be driven by factors such as: organization requirements; organization capability; the activities performed by the organization; available resources; threats and vulnerabilities; and the risk tolerance and appetite of stakeholders. However, certain choices can be limited, as the organization creating the ledger and the initial record (e.g. genesis block in a blockchain system) will set certain parameters (such as the encryption algorithm) that all participants in a DLT system have to follow.

6.3 Identity

The importance of identity in Distributed Ledger Technologies (DLT) results from the fact that transactions can include identifiers relating to a natural person, a legal entity, a thing or a process.

One could rely on traditional, trusted third party entities to provide these digital identifiers to secure communications and transactions (X.509 Certification Authorities, DNS registrars, etc.), but decentralized identifiers put the control back in the hands of the identity owner, which is an important consideration for self-managed, decentralized identity.

6.4 Privacy

6.4.1 General

Some uses of DLT systems necessarily involve the acquisition and storage of PII by the system. This raises the question of how PII is stored by the system. A significant characteristic of DLT systems is that information stored in the distributed ledger is normally immutable - it cannot be altered once it is written to the ledger. This characteristic can clash with some of the principles that apply to the

protection of PII; in particular, it might not be possible to enable the PII principal to delete any of their PII once written to the distributed ledger. A PII principal can edit the PII using another transaction. However, previous information will remain on the ledger and might be accessible.

There are many techniques to enhance privacy with technology. In particular, Privacy Enhancing Technologies offer an ability to protect data through a variety of means. Some of the current approaches include the following:

a) Cryptographic techniques;

b) Network techniques;

c) Privacy frameworks;

d) Channel-based techniques.

See ISO/TR 23244 for more details on privacy protecting techniques.

Whether use of these approaches is acceptable in relation to PII or not depends strongly on the use case and the legal basis for the storage of the PII. In some cases, it can be required by law to maintain a permanent and unchanging record of some PII - an example is a land registry. In other cases, it can be required to enable the PII principal to delete their PII. Since DLT does not support this, PII should therefore not be stored on the ledger. Instead, the PII needs to be placed into an (mutable) off-ledger database with some kind of pointer to this PII being placed into the ledger records. This approach enables the PII to be updated and/or deleted while preserving the immutability of the ledger records.

When designing and operating DLT systems, the guidance and principles stated in ISO/IEC 29100 can be applied; there can be further considerations of the issues specific to DLT systems.

DLT architectures will have a major impact upon privacy and the protection of PII. Within an architecture, the implementation of access management, security and cryptographic services to applications, APIs and storage across both DLT and non-DLT systems can protect PII through technical and non-technical means. Many information security controls, which protect the confidentiality, integrity and/or availability of information and data are also suitable for the protection of PII. The selection of controls can be made using risk management techniques, business and information security policies and the capabilities of the services in the architecture. ISO/IEC 27001, ISO/IEC 29134, ISO/IEC 29151 and ISO/IEC 29100 provide controls and guidance for general security and privacy aspects

It is also plausible to use the “privacy by design” paradigm in DLT development to produce DLT systems that provide appropriate PII protection.

PII can plausibly be contained in 2 datastores:

— on-ledger;

— off-ledger.

6.4.2 On-ledger PII storage

PII can be stored within DLT records (e.g. blockchain blocks). If it is required for PII to be modified or deleted, for example to comply with legal or regulatory requirements, in many DLT systems, it might not be possible in practice to modify or delete those records. Worst-case options might include a hard fork. In the case of some private DLT systems, the owner might be forced to shut down the system entirely. Also, there is a continuing risk of exposure, for example via leaked keys or preimage attacks, even if the PII is encrypted or hashed. Criteria for good encryption algorithms and parameters are discussed in ISO/IEC 18033-1, but good encryption is not necessarily adequate for protection of PII in a DLT system.

As the ledger increases in size and as transactions are added, the accumulation of data within the ledger itself and links to external databases and storage can lead to aggregation effects, potentially enabling direct or indirect identification of a PII principal. Advances in analysis and profiling capabilities can also lead to aggregation or other effects, again increasing the risk of identification of a PII principal.

6.4.3 Off-ledger PII storage

Where data is held off-ledger, privacy and PII protection can be addressed by adopting the approach of ISO/IEC 29100.

DLT systems typically use hashes of files to allow the actual data to be held off-ledger whilst a validated record of the file is held on the ledger. This facilitates large, related files to be held off-ledger, whilst the integrity of the data referenced is maintained through use of the hashing function on the data. Identifiers can be used to point to PII held off the ledger, where these identifiers are not derived from the data itself and will probably only have a one-way relationship.

Storing information "off-ledger" can provide confidentiality of the transaction details. The off-ledger system can be configured to restrict access to the transaction details to authorized parties only, supporting use cases where participants wish to keep details of their transactions private from other DLT participants.

However, storing information "off-ledger" negates many of the advantages of using a DLT system in the first place. Although the use of hashes allows to ensure integrity and detect tampering, without transaction details, the blockchain can no longer be a single, shared record that can be verified by multiple parties according to a set of rules ("source of truth"). For example, the issuance and trading of negotiable, fungible digital assets are no longer possible without reference to an off-ledger position-keeping system (i.e. a system that monitors the quantity of assets recognised at the DLT system level and its compliance with the transactions of which the system is aware).

Additionally, storing transaction information off-ledger typically requires that both counterparties maintain their own record, or delegate that responsibility to a Trusted Third Party, which brings with it the same costs and disadvantages as restricting read access to the blockchain.

A major challenge for DLT systems is the ability to ensure that a file or a block on a node and any associated off-ledger storage has been deleted.

DLT can also integrate with distributed file systems, for example BigChainDB and IPFS. These file systems also have the same challenge of ensuring that if a file is deleted, confirmation is received that on each node the file was deleted.

6.5 DLT Governance

See 3.7. Directing and controlling implies creating a framework for setting policy, decision rights (including access), accountability, and ongoing enforcement in the DLT systems. DLT systems governance is challenging due to the distributed and decentralized nature of most DLT systems because of the following:

— Multiple owning stakeholders (individuals, organizations) might need to be considered;

— Governance across organizational boundaries or jurisdictions might occur;

— Governance considerations for setup of DLT systems and ongoing operations might be different;

— A high degree of centralized decision rights might be needed initially to enable distributed and decentralized control later;

— The specification and execution of decision rights might need to be organized to operate in a distributed and decentralized environment;

— Incentive mechanisms might need to be implemented that work in a distributed and decentralized environment;

— Records are decided upon through distributed and decentralized consensus mechanisms;

— Transaction parameters are primarily defined using distributed and decentralized mechanisms;

— Accountability should be explicitly specified reflecting the distributed and decentralized nature;

— Governance might need to scale across a large number of nodes and stakeholders;

— An authorization functionality might be required to support multiple parties across organizations and sometimes across borders;

— An appropriate identity management system might need to be in place.

Specific decisions that will have to be resolved might include selecting and defining consensus mechanisms, how voting mechanisms work, how conflicts are resolved, how mining works, and how organizational and human agents participate.

Mechanisms and monitors need to be defined on operational DLT systems such that they can verify that the governance policies are being enforced and can signal when governance and decisions need to be adjusted.

ISO/TS 23635 provides further guidance on governance for DLT systems.

6.6 Management

Management of DLT systems includes the management of each node in the system plus all the challenges of managing distributed systems consisting of many nodes. It also includes issues related to the differing ownership and control of each of the nodes, the scale of the systems, and the handling of the consensus mechanism.

Management covers the complete lifecycle of the DLT system: development, testing, configuration, deployment, operations, monitoring, maintenance, and decommissioning.

Governance needs to be in place to support all types of management, and management needs to report exceptions and metrics to any governance systems as appropriate. Due to the distributed and decentralized nature of ownership, identifying who is responsible for executing governance and management tasks needs to be carefully considered and responsibilities allocated appropriately.

Targets for management in DLT systems can include

— nodes (including the code of the DLT platform),

— distributed ledgers,

— transactions,

— secure runtime environments,

— smart contracts,

— cryptography,

— channels,

— membership,

— users (clients),

— data storage,

— event systems,

— applications,

— compliance with regulation,

— networks, and

— security – permissions and security for nodes, members, smart contracts, and users (see 6.2).

Management for each target will be different.

Managing configuration includes the initial configuration for deployment and updating the configuration on an ongoing basis in one or many nodes, smart contracts, ledgers, applications, or users.

Deployment management includes the deployment and disposal of nodes, networks, smart contracts, ledgers, databases and applications across a distributed network of physical systems, virtual systems and DLT systems. Scale and dependencies can be challenges here.

Operations management includes

— starting nodes and applications,

— stopping nodes and applications, and

— changing network connections.

Monitoring includes

— monitoring performance of nodes and communication networks,

— monitoring status and availability of nodes and communication networks, and

— monitoring business specific metrics for governance.

Maintenance includes implementing and deploying ongoing changes to any element of the DLT systems, networks, and applications. Getting changes deployed across thousands of nodes in a timely manner without causing disruption in availability and interoperability of APIs might be a challenge.

Disposal includes decommissioning or deleting any element from the DLT system. Doing this without causing disruption with the DLT system is important.

Management and governance of DLT systems need to be carefully coordinated. As an example, the governance process can decide that the code of the DLT platform needs to be updated, say to fix a security issue – the management process needs to ensure that all nodes in the DLT system are updated accordingly, or else the security issue might remain.

Software for DLT platforms can be updated to fix bugs and change rules, including the consensus mechanism, the communication protocol between DLT nodes, data structure, size of block, and difficulty for Proof of Work and so on. There are two different types of migrations regarding software update called hard fork and soft fork.

A hard fork is a change that does not keep backward compatibility. It requires all DLT nodes to update to the new version of the software. Non-updated DLT nodes cannot accept transactions or ledger data that are created by updated DLT nodes according to the new rules. Therefore, non-updated DLT nodes are not able to maintain a connection to the network of updated DLT nodes.

Hard forks can cause permanent divergence of ledgers between non-updated DLT nodes and updated DLT nodes. This kind of divergence is called a "ledger split" or "fork". Hard forks can be caused either deliberately or accidentally.

A soft fork is a change to keep backward compatibility. It is a way to encourage DLT nodes to update to the new version of software. Non-updated old DLT nodes and updated new DLT nodes coexist on the same DLT network. Old DLT nodes can accept transactions or ledger data that are created by new DLT nodes, but on the other hand, new DLT nodes can reject those created by old DLT nodes. If a majority of DLT nodes update to the new software, it is expected that other old DLT nodes will also update since their transactions and ledger data can be accepted by a larger number of DLT nodes.

6.7 Interoperability

DLT systems are being proposed for a wide variety of purposes where DLT systems will need to work together both with other DLT systems and with non-DLT systems. Looking at the evolution of credit

cards and of electronic data interchange (EDI), DLT systems representing crypto-assets will need to interoperate with other crypto-asset systems. Importing information to and exporting information from DLT systems will be facilitated by common information formats and semantics. This is also true when interoperating as part of a supply chain information exchange.

Interoperability is to be differentiated from portability, interchangeability, or uniformity. Portability implies the ability to migrate applications or information from one system to another, rather than to exchange information. Interchangeability implies being able to replace one system with another to fulfil the same requirements. Uniformity implies sameness of design. Interoperability focuses on exchange of information, where the information can be efficiently exchanged so that different systems can work together.

There are various perspectives of interoperability that need to be considered. These perspectives are called "facets". There is the 5-facet model, which is described in detail in ISO/IEC 19941. The 5 facets are:

1. Transport Interoperability (see 3.23)

   DLT is often thought of operating in an Internet-based environment; however, DLT can operate in other networking environments as well. The transport facet deals with the communications infrastructure – how to get bytes of data from one system to another.

   This includes the networking hardware: Ethernet, Wi-Fi, Broadband (xDSL), Satellite, etc. It also includes the transport protocol used: HTTP(S), AMQP, MQTT, Kafka, etc.

   If the transport facets do not align, then some form of protocol conversion is likely to be required.

2. Syntactic Interoperability (see 3.22)

   The focus is on the "syntax of the data". The syntax of the data is ideally the same for the source service and the target service. However, if the syntax does not match, e.g. the source uses JSON syntax but the target uses XML, it is possible to map the data using commonly available tools. EDI environments have leveraged numerous standardized syntaxes.

3. Semantic data Interoperability (see 3.20)

   Semantics concerns itself with concepts within a domain, their properties, terminology and vocabulary and the structural relationships between them. Semantics also concerns itself with the correctness, validity and completeness of data.

   The focus is on the "semantics of the data". The semantics of data are commonly expressed by an ontology. Compatible ontologies simplify the porting of data between source and target services. If the ontologies are incompatible, additional resources can be applied to detect inconsistencies. These inconsistencies might be resolved or semantic data interoperability requirements could be relaxed to enable the data to be ported.

4. Behavioural Interoperability (see 3.3)

   It needs to be specified how the service interface uses the exchanged data with related conditions and constraints

5. Policy Interoperability (see 3.16)

The 5-facet model is also used in ISO/IEC 21823-1. These 5 facets apply to understand interoperability in widely distributed systems such as cloud and DLT systems.

In addition to the five facets, it is necessary to understand what systems or applications are involved in DLT systems – what entities are communicating and what is the nature of their communication.

When considering interoperability for digital assets, it is important to ensure the concept of inherited characteristics.

When considering interoperability, 4 interactions between entities in DLT systems are relevant, as shown in Figure 2, which removes the details of entities entirely within the DLT systems.

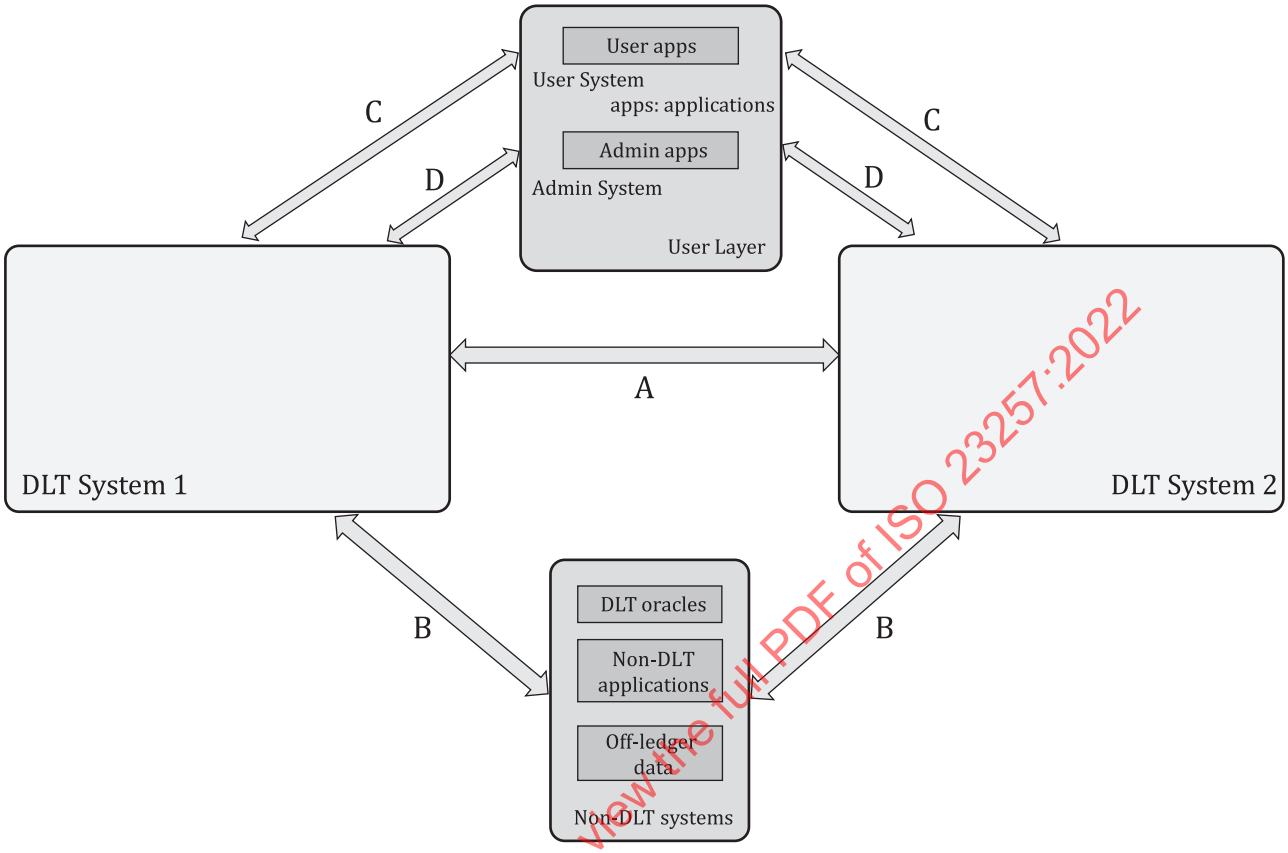


Figure 2 — Interoperability interfaces – DLT user system, DLT admin system

Figure 2 identifies four potential interfaces where interoperability might apply:

A – directly between 2 (or more) DLT systems;

B – between DLT systems and external non-DLT Systems;

C – between user applications and DLT systems;

D – between admin applications and DLT systems.

Examples of interoperability among many systems include:

— System 1 interoperates with System 2 with a subset of the functionality and properties common to the two systems;

— System 2 and System 3 interoperate with the functionality and properties common to the two systems;

— System 1 and System 3 can interoperate with a subset of the functionality and properties common to the three systems.

DLT discovery supports the interoperability between DLT systems. Discovery includes the activities that a DLT system performs to share information about what it is and does. A DLT system can communicate this information, for example, using a service directory or catalogue, to other entities to facilitate their participation or to exchange data and services. An optional Ledger Name Service (LNS) can be implemented in a DLT system analogously to the Domain Name Service (DNS) used in the Internet to identify the network connection of a ledger service. This could enable cross-ledger applications to

conveniently inter-operate. A naming convention could also be standardized. Detailed definitions for discoverability concepts and relationships are outside the scope of this standard.

It can be useful to facilitate the automation of making different DLT systems interoperable, as opposed to needing to find, evaluate, connect and map information between the systems on a bespoke basis every time, a means by which DLT systems can advertise their discovery information so other systems can determine compatibility and relevance is useful. Standardization of metadata taking into consideration the 5-facet model can be useful to enable the interpretation of discovery information.

Some examples of DLT system discovery information could include: processes and purposes, information and services that can be offered or consumed, security, assurance, access, cost, advertisement, operational information, governance requirements, non-functional characteristics, regional and legal restrictions, and other useful attributes to facilitate manual or automated integration.

Different integration scenarios could need different information during discovery, integration or participation, i.e.

— external applications seeking to find and understand the capabilities, requirements, and reputation of an existing DLT systems and services as a user, and

— prospective nodes seeking a ledger of a DLT system, where the existing counterparts are fewer, and the vocabulary are less established.

6.8 Data flow

Understanding data flow (see 3.5) is essential to protect the privacy and confidentiality of data in DLT systems.

On a system level, DLT system data flows can be divided into internal flows and external flows. There are five fundamental data flows relative to DLT systems as shown in Figure 3:

— Data flow Z: data flowing among the nodes of the DLT system;

— Data flow A: data flowing between two separate DLT systems when they interoperate;

— Data flow B: data flowing between a DLT system and non-DLT systems connected to it;

— Data flow C: data flowing between administration applications and a DLT system;

— Data flow D: data flowing between user applications and a DLT system.

However, data flows in real world applications are often complicated and difficult to understand. The data flows in DLT systems are often triggered by the data-related operations of stakeholders. In particular, DLT data can flow between systems that belong to or are related to different stakeholders. Meanwhile, the use of devices is common in DLT systems, involving a variety of devices such as portable devices and IoT devices. In addition, in some cases, cloud computing services are used to provide capacities such as storage, which makes data flows in DLT systems more complex.

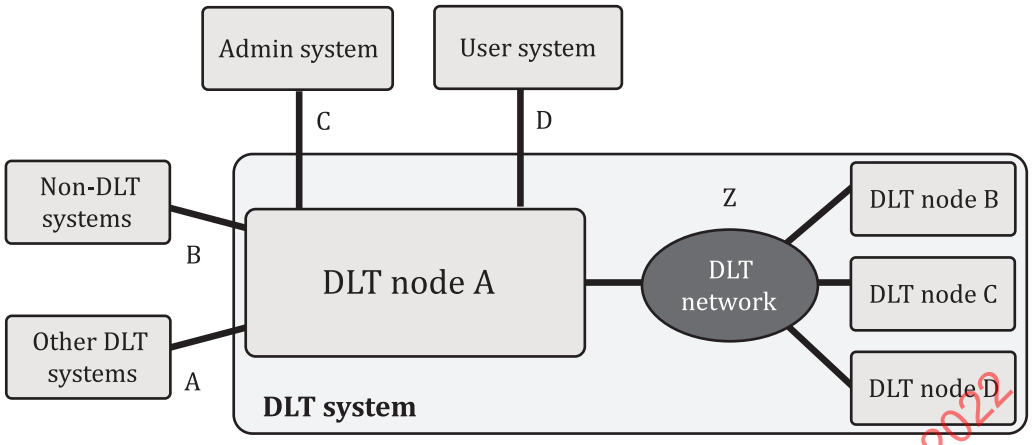


Figure 3 — DLT data flows

7 Types of DLT systems

The distinction between public DLT systems and private DLT systems is about who can read transaction records on the ledger. A public DLT system has transaction records that are readable by anyone (i.e. by the general public). By contrast a private DLT system has transaction records that are readable by not just anyone - read access to the transaction records is deliberately limited - the records are private to the extent that read access is limited to some particular group.

Only read access to the transaction records is addressed here. It can be the case that adding transaction records to the ledger is also open to all, or it can be the case that adding transaction records to the ledger is limited to some particular group. Both are possible for public DLT systems.

The phrases "permissionless DLT system" and "permissioned DLT system" are about whether it is necessary to have permission to use the system (permission implies some form of authentication and authorization for the user concerned). A permissionless DLT system requires no permission in order to use or to operate the system. A permissioned DLT system requires permission in order to use some operations of the system. “Operations” here includes the basic DLT functions such as “read transaction records”, “add transaction records”, and “verify transaction records.”

There are four combinations of these concepts to understand:

1) public permissionless DLT system

This is a DLT system that is open to anyone to use and no permissions are required to do any operations on the system, including participating in consensus mechanisms and verifying transactions. This is one of the classic cases for blockchain - Bitcoin and Ethereum are instances of these.

2) public permissioned DLT system

This is typically a case where read access to the transaction records is open to anyone, but where other operations on the system require permissions of some kind, e.g. anyone can read but only specific people can add transaction records.

An example of such a DLT system would be a system enabling consumers to verify the provenance of items of food - they can read the transaction records of the history of the item ("from farm to fork") but only farmers, shipping companies, etc. would be enabled to create such transaction records.

3) private permissioned DLT system

This is a classic and common case where all operations on the DLT system are limited to specific groups and where permissions are required to perform any operation on the system, even reading the transaction records. Permissioned DLT could ensure visibility of the different owners of the nodes

that help run the DLT network. Such cases can be handled by using open-source DLT systems that are permissioned. Hyperledger Fabric is an example of private permissioned DLT system.

4) private permissionless DLT system

In this case, operations on the DLT system are limited to specific groups, but where no permissions are required to perform those operations - this can be done using "other means" to limit participation, such as installing the DLT system on a private network.

8 Architectural considerations for DLT Systems

8.1 Characteristics and relationships

To understand DLT systems, it helps to identify a limited set of characteristics and understand their relationships to each other as shown in Table 1. The technologies are categorized according to the following four characteristics:

— Ledger storage architecture;

— Ledger control architecture;

— Ledger subsetting;

— Ledger permissions.

Very few solutions will live entirely "on-chain", so it is also important to recognize the components that surround the ledger in an end-to-end solution. A ledger is deployed to infrastructure – computation, network, and storage need to be set up to handle ingestion of the messages. In some systems, certain types of data such as media (image, video, audio) are not put on the ledger and instead go into separate storage. Other data might be inappropriate to put on the ledger because the data, e.g. PII, are governed by data protection regulations, such as the European Union's GDPR, and immutability and transparency are seen as negatives. As a result, this data will typically reside in an off-ledger data store.

In the context of DLT systems, storage architecture is different from control architecture (e.g. consensus) and ledger storage is different from data content storage (e.g. media data). As a result, characteristics associated with ledger storage architecture and ledger control architecture can be separated as follows.

NOTE Centralized ledger technologies are included in Table 1 and the following sections for completeness and understanding. Centralized ledger systems are outside the scope of this document.

Table 1 — Typology of DLT Systems based on a limited set of characteristics

Ledger technology	Ledger storage architecture	Ledger control architecture	Ledger subsetting	Ledger permissions
Centralized Ledger Technologies (i.e. traditional ledger)	**Centralized (ledger storage) architecture** (i.e. central storage)	**Centralized (ledger control) architecture** (i.e. central control)	**Complete replica of the ledger** (i.e. full duplication)	**Permissionless or Permissioned**
Distributed Ledger Technologies (DLT)	**Distributed (ledger storage) architecture** (i.e. distributed storage)	**Decentralized (ledger control) architecture** (i.e. decentralized control, based on a consensus mechanism)	**Subset of the ledger** (i.e. partial replica)	
			Complete replica of the ledger	

Transaction performance is an important consideration and is influenced by the architecture of the network.

8.2 Ledger technology

Ledger technologies are designed to support and implement ledgers. They can be **centralized** or **distributed**.

8.3 Ledger storage architecture

A ledger storage architecture can be:

— **Centralized** (ledger storage) architecture has a central server (broker) which stores a single complete instance of the ledger;

— **Distributed** (ledger storage) architecture is where each node might store a full or partial replica of the distributed ledger.

8.4 Ledger control architecture

A ledger control architecture can be:

— **Centralized** (ledger control) architecture is where a central server (or authority) controls the decision making relating to the distributed ledger (i.e. the validation of the new blocks of records);

— **Distributed** (ledger control) architecture is where all architecture elements (particularly nodes in the DLT system) control the decision making relating to the distributed ledger, based on a consensus mechanism run by the distributed system (see ISO 22739:2020, 3.32);

— **Decentralized** (ledger control) architecture is where multiple nodes (or authorities) in the DLT system control the decision making relating to the distributed ledger, based on a consensus mechanism running as a decentralized system (see ISO 22739:2020, 3.19).

Ledger control is associated with ledger consensus mechanism/model. The latter can be described within the DLT core software (or DLT client software).

8.5 Ledger subsetting

In the context of DLT systems, ledger subsetting can be:

— **Subset of the ledger** (i.e. partial replica);

— **Complete set of the ledger** (i.e. full replica).

8.6 Ledger permission

The types of permissions applied to ledgers are **permissionless** or **permissioned**.

9 Architectural views of reference architecture

9.1 General

9.1.1 Five architectural views

DLT systems can be described using a viewpoint approach. Five distinct views that can be used to describe a reference architecture are:

— User view;

— Functional view;

— System view;

— Implementation view;

— Deployment view.

Figure 4 shows the possible transformations between them, and Table 2 provides a description of each of these views.

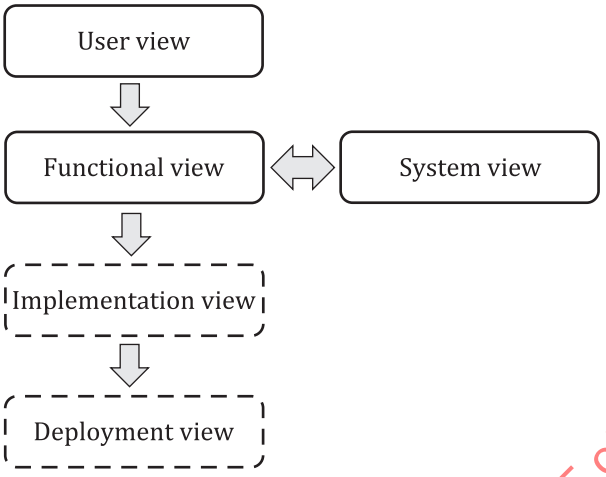


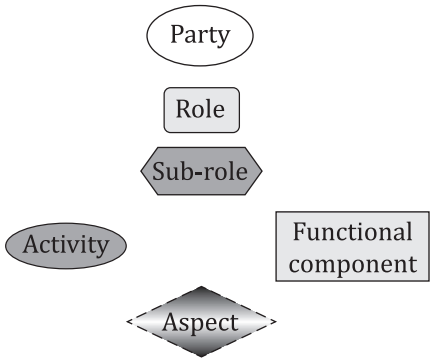
Figure 4 — Transformations between architectural views

Table 2 — Reference Architecture Views

RA view	Description of the RA view
User view	The system context, the parties, the roles, the sub-roles and the activities of DLT system
Functional view	The functions necessary for the support of activities of DLT system
System view	The alternative view of functional view that emphasizes the relationships of the various components and their relative placement within the system
Implementation view [a]	The functions necessary for the implementation of DLT system within service parts and/or infrastructure parts
Deployment view [a]	How the functions of DLT are technically implemented within already existing infrastructure elements or within new elements to be introduced in this infrastructure
[a] The Implementation and Deployment views are outside the scope of this document.	

9.1.2 Notation of diagrams

The following conventions apply: Diagrams are used throughout this document to help illustrate this reference architecture. Figure 5 provides the conventions used regarding the content of the diagrams.



NOTE "Aspect" is understood as referring to "Cross-cutting aspect".

Figure 5 — Legend to the diagrams used throughout this document

The reference architecture in this document uses the term "ICT" and "ICT systems". This term is used to make it clear that the reference architecture covers not only the compute and storage technologies associated with computer systems, but also the communication networks that link systems together.

Parties in a DLT system are its stakeholders. A party can assume more than one role at any given point in time and can engage in a specific subset of activities of that role. Examples of parties include, but are not limited to, individuals, businesses and governments.

Different sub-roles can share the DLT activities associated with a given role. Descriptions of the DLT roles and sub-roles are provided in 9.2.

9.2 User view

9.2.1 General

This document describes a set of roles and sub-roles which address the main activities associated with DLT systems. As shown in Figure 6, the roles and sub-roles of DLT systems are the following:

- DLT users;
- DLT administrators;
- DLT providers:
  - DLT System operators;
  - DLT Node operators;
  - DLT Application operators.
- DLT developers:
  - DLT application developers (for DLT users and DLT providers);
  - DLT system developers (for DLT providers).
- DLT governors;
- DLT auditors.

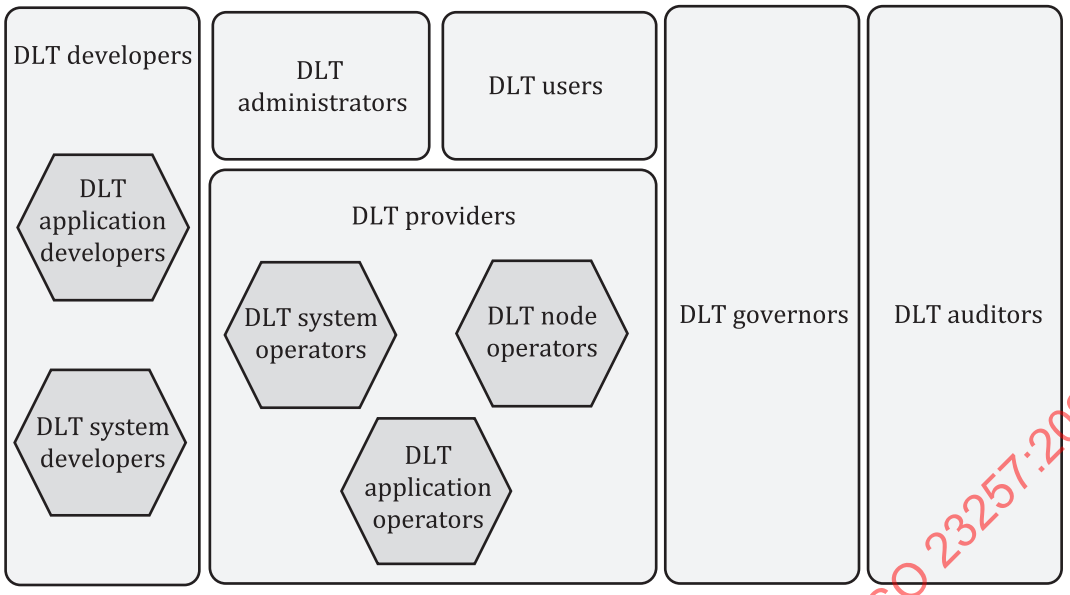


Figure 6 — DLT System Roles and Sub-roles

To recognize the responsibility boundary in a DLT system, it is needed to understand not only role and sub-role but also activities in detail. Each activity is shown in the following sub-clause of role and sub-role.

Not all of these roles/sub-roles necessarily exist for all DLT systems. It is also important to recognize that in some cases one person or one organization can perform multiple roles/sub-roles. It is also the case that different roles/sub-roles can be split across different organizations.

NOTE All activities need to be carried out under the premise of protecting privacy.

9.2.2 DLT users

A DLT user is a role which uses a DLT solution. This can represent an individual, organization, device or system.

It is typically the case that DLT users use a DLT system by means of a DLT application or off-ledger code that interacts with the DLT API, rather than directly interacting with a DLT node. This is true for DLT users which are automated systems rather than human users, where the interaction would typically occur via an API offered by the DLT application.

Activities include the following:

— Using DLT user applications – (interact with business systems appropriately);

— Installing user applications;

— Configuring DLT user applications;

— Installing client or application for interacting with a DLT system;

— Handling exceptions or failures.

9.2.3 DLT administrators

DLT administrators perform specific administrative activities (in particular for security configuration). The scope of administrators will be different in different systems and networks – some will be limited to DLT nodes and some will be able to act for DLT solutions.

Activities include the following:

— Managing security policy;

— Managing cryptographic keys;

— Handling exceptions;

— Installing user applications;

— Configuring DLT user applications and administration applications;

— Managing role-based access controls for smart contract creation and deployment and invocation of smart contract.

9.2.4 DLT providers

9.2.4.1 General

DLT provider is a role that may own and operate one or more nodes within DLT systems and DLT networks. DLT providers agree to create/instantiate nodes, join networks, pay for, and handle legal agreements for joining the network. This role is the business stakeholder.

Sub-roles include the following:

— DLT System operators;

— DLT Node operators;

— DLT Application operators.

9.2.4.2 DLT system operators

DLT system operator is a role that administers and operates the physical and virtual systems and networks that host and run the components of the DLT systems and platforms in the DLT nodes. It handles network and multiple nodes connections, interoperability between nodes, and enforces policy between nodes.

Activities include the following:

— Managing communication networks for the DLT system;

— Deploying the physical and virtual systems;

— Establishing environments and processes.

9.2.4.3 DLT node operators

A DLT node operator is a role that administers and operates one or more nodes within a DLT system for a DLT provider. Node operators are responsible for the lifecycle of the node which could include business responsibilities, preparation for deployment, running nodes, managing and maintaining nodes.

Activities include the following:

— Handling business concerns: including customer relationships and financial processes, and business statistics;

— Preparing for deployment: including defining deployment processes, designing system continuities, and selecting audit requirements;

— Running and operating: including preparing systems, implementing deployment processes, running ledgers, smart contracts and consensus mechanisms in secure runtime environments, implementing system continuity and responding to user requests;

— Managing, securing and operating: including operating and monitoring systems, recovering nodes, managing assets and managing security and risk;

— Maintaining that DLT systems are updated appropriately.

NOTE Operator can be directed by Regulator or Governors (depending on scope)

9.2.4.4 DLT application operators

DLT application operator is a role that administers, owns, operates and maintains DLT applications systems to provide DLT services for DLT users directly or indirectly.

DLT application operator may provide DLT services by owning or operating a DLT node, or may provide DLT business services through services provided by a DLT node owner.

Activities include the following:

— Providing, monitoring and managing DLT business services;

— Business management;

— Privacy protection;

— Managing security and risk;

— Obtaining audit report;

— Problem handling;

— Ensuring compliance.

9.2.5 DLT developers

9.2.5.1 General

DLT developer is a role that creates and maintains the code and specialized equipment for any part or implementation of the DLT systems or applications. There are two sub-roles for DLT developers - DLT application developers and DLT system developers.

9.2.5.2 DLT application developers

DLT application developer is a role that creates and maintains DLT applications and smart contracts that run in conjunction with DLT systems. It can also involve applications which run outside the nodes of the DLT system, interacting with them by means of a User API offered by the nodes. DLT applications can be used, hosted and run by the DLT users or DLT providers.

Smart contract development sometimes needs special tools and skills since a smart contract often represents a business process which can range from as simple as a basic transaction between two parties to very complex scenarios in areas such as financial services or supply chain. It is important that only smart contracts that follow approved business process flows can be deployed to the DLT stack in a given environment.

The methods in these smart contracts represent actions that can be taken within a contract by identities/identifiers assigned to domain-specific roles, e.g. owners, buyers, etc.

Organizations could allow only certain identities/identifiers to deploy applications and new instances of a smart contract to their ledger implementations. Smart contract creation can be governed at the application level, the DLT platform level, or the smart contract level.

DLT application and smart contract developers need to take into account roles and access requirements for them.

Activities include the following:

— Design, create, integrate and maintain business systems based on the DLT systems;

— Design, create and maintain components or smart contracts in the DLT systems.

9.2.5.3 DLT system developers

DLT system developers play roles that develops the physical and virtual systems that host and run the components of the DLT systems and platforms in the DLT nodes.

Activities include the following:

— Developing DLT platform components;

— Testing components of DLT systems for node operators and node users.

9.2.6 DLT governors

DLT governors play roles which perform governance of the entire DLT systems (and networks), but not necessarily the entire DLT solution. The DLT systems can be considered infrastructures while the DLT solutions built on it are considered business applications. Both would have governance but can (and often are) governed by different governing bodies. Governance of decentralized systems can require new governance structures and the ability to change over time. In some DLT systems, governance is performed not by one governing body, but instead by a collective of interested stakeholders.

There is a need for a role that governs the DLT systems as a whole and keeps the DLT systems able to execute the tasks for which they were established.

Activities include the following:

— Developing DLT policy considering applicable laws and regulations;

— Communicating the policy with stakeholders;

— Resolving conflicts and managing changes;

— Defining polices for consensus mechanisms;

— Defining policies for nodes that can participate in the DLT networks– including the minimum security requirements;

— Working with DLT providers;

— Working with DLT node operators to ensure monitoring and governance is enforced.

9.2.7 DLT auditors

DLT auditors ensure that policy, governance and regulation are adhered to in DLT systems. They can work with operators, regulators, governors, etc.

Activities include the following:

— Collecting evidence for audit (to satisfy selected requirements, criteria, frameworks or options);

— Performing DLT systems and DLT applications audit;

— Reporting audit results.

NOTE 1 Audit-related activities need to be carried out under the premise of protecting privacy.

NOTE 2 There are many types of audits, optional and required that could apply here.

9.3 Functional view

9.3.1 Functional categorization framework

The functional view is a technology-neutral view of the functions necessary to form a DLT system. The functional view describes the distribution of functions necessary for the support of activities in DLT systems. The functional architecture also defines the dependencies between functions. The functional view addresses the following DLT system concepts: functional components; functional layers; and cross-layer functions. Figure 7 illustrates the concepts of functions, layers and functional components.

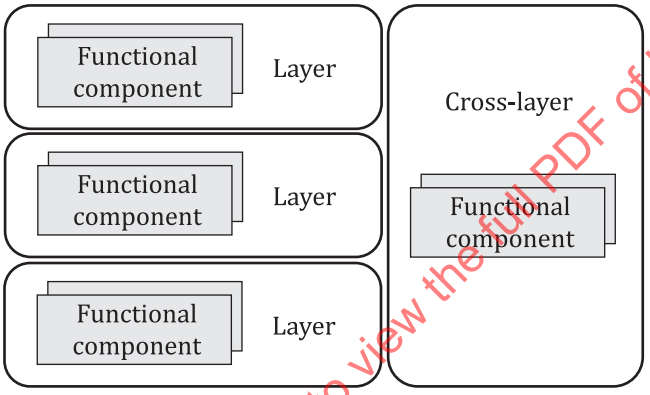


Figure 7 — Functional layering

The hierarchical framework of the reference architecture of DLT is shown in Figure 8, Functional components.

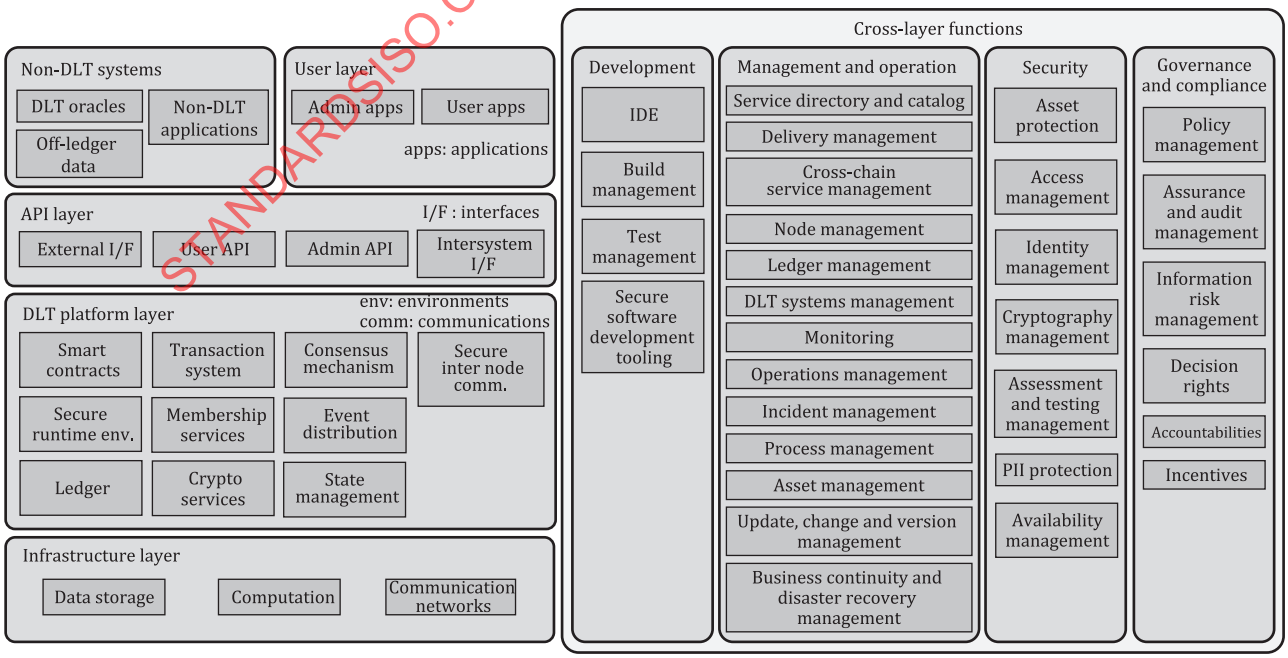


Figure 8 — Functional components

Figure 8 consists of non-DLT systems and four layers, as well as a set of cross-layer functions across the layers. The four layers are the following:

— User Layer,

— API Layer,

— DLT Platform Layer,

— Infrastructure Layer.

9.3.2 Non-DLT systems

The non-DLT systems layer contains systems outside the DLT system that a DLT system communicates with in order to accomplish its business goals. This includes the following:

— DLT oracles

DLT oracles represent trusted systems designed to supply external data to a DLT system or to respond to events from DLT systems. Transformation logic and services can be used to perform the translation needed to communicate data to and from DLT systems. DLT oracles can enable smart contracts to ingest real world data during code execution.

— Non-DLT applications

Non-DLT applications are any applications outside the DLT system with which the DLT system communicates, either to send or receive data. Whether reporting and analytics or machine learning, there are multiple places where analysis is done on ledger data. Such an application can provide data and services to DLT applications via an application programming interface, or it can ingest data from a DLT to provide ancillary services.

— Off-ledger data

Off-ledger data is any data stored outside the DLT system that can hold data that relates to the DLT system in some way. An example might be a database holding additional data relating to transactions held on the ledger.

9.3.3 User layer

User layer contains functions to enable DLT customers to interact with other layers and cross-layer functions.

User applications are applications that run separately from the DLT systems. They act as clients to the DLT systems, and are used by users, usually to perform domain-specific or application-specific operations.

Administration applications are applications that run separately from the DLT systems that act as a client to the DLT system, used by administrators supporting capabilities to maintain and/or update applications and systems.

NOTE DLT user and DLT administrator roles both leverage the user layer. See 9.2 User view for more understanding on tasks that roles and users/operators/administrators perform and need APIs for.

9.3.4 API layer

The API layer contains functions that provide reliable and efficient access to the DLT system for applications, users and non-DLT systems, by calling functional components in the DLT Platform layer, and thus access to the underlying services of the platform layer and the cross-layer functions.

— **External interfaces** – off-ledger access services provide secure means to communicate between the DLT system and the outside, non-DLT systems;

— **User API –** application programming interface that provides access to domain specific functions;

— **Admin API** – application programming interface that provides access to administrator and operator functions;

— **Intersystem interfaces** – a node in a DLT system can communicate with other node in a separate DLT system using Intersystem interfaces.

9.3.5 DLT platform layer

The DLT Platform layer contains the core functions of the DLT systems that can run in a DLT node. It also involves communication between nodes.

Key capabilities include the following:

— Consensus mechanisms which coordinate the data and DLT account records in the ledger between nodes;

— Communications between DLT nodes and DLT systems via events and secure protocols;

— Cryptographic services including encryption, digital signature and other capabilities to ensure security compliance and tampering resistance of DLT systems;

— Smart contracts running in a secure runtime environment can automatically implement scheduled logic that can be added selectively according to different requirements of the applications;

— Functions to enable increased performance by means of efficient caching, reliable storage, load balancing.

The DLT Platform layer connects hardware or network infrastructure provided by the Infrastructure layer, to relevant functional support services in the API layer. The capabilities supported by the DLT Platform Layer include

— Smart contract,

— Secure runtime environments,

— Ledger,

— Transaction system,

— Membership services,

— Cryptographic services,

— Consensus mechanism,

— Event distribution,

— State management, and

— Secure inter-node communications (networking).

To describe these capabilities further:

— **Smart contract**

A smart contract is a computer program stored in a DLT system wherein the outcome of any execution of the program is recorded on the distributed ledger. Smart contracts expose programmatic functionality to users of the DLT systems. Users interact with smart contracts by initiating transaction events which reference the contract address.

In public DLT systems, smart contracts are almost exclusively distributed in nature, in private DLT systems, they can be distributed or centralized. When centralized, there are a number of trust elements associated with centralized smart contracts, e.g. running each in a separated trusted execution environment to satisfy the parties involved in transactions in the network. Smart contracts represent more the process of agreeing on an outcome than its legal status. The outcome can be legally binding.

NOTE Smart contracts (also called chain code, programmable asset or programmable contract) are instantiated as computer programs that execute in a secure runtime environment when a user sends a transaction of a particular type to the DLT system.

— **Secure runtime environments -** During runtime, a transaction can invoke smart contracts, which require a secure execution of code. A secure runtime environment ensures verifiably secure execution of code. A secure runtime environment can leverage both software and hardware security solutions. An example is the use of a secure container that contains a set of signed runtime components such as a secure operating system, libraries for DLT-supported programming languages, their respective runtimes, and the like.

— **Ledger**

A ledger (see 6.1) is an information store which keeps a final and definitive record of transactions.

A ledger supports data storage capabilities. Data Storage capabilities support writing and querying of various types of data generated during the operation of the DLT system, such as the ledger, transaction information, etc. Technical implementation can be relational database, key-value pair database, file database, and so on. It needs to support data fragmentation and routing capabilities where database sharding is supported.

— **Transaction system**

The transaction system is the component that manages the addition of transaction records to the ledger. A transaction system can contain a temporary data area to store unconfirmed transactions, called a transaction pool. After a transaction is validated, it is put in the transaction pool, if valid. Transactions to be recorded in the ledger are chosen from the transaction pool. Once transactions are confirmed to be recorded in the ledger, they can be removed from the transaction pool. Each DLT node can have its own transaction pool.

— **Membership services**

The membership services are services that manage the identity, privacy, confidentiality and auditability within the DLT system. Membership services only apply to permissioned DLT systems.

— **Cryptographic services**

The cryptographic services component provides the DLT system with access to the necessary cryptographic algorithms, either directly or by providing an interface to hardware or software that implements the algorithms. Hash functions and digital signatures are examples of algorithms that are commonly used in DLT systems. Hash functions are often used to protect the ledger from modifications. Any change to information in the ledger will result in a computed hash that is different from the hash that was previously computed and stored for the ledger. A new hash is computed each time a transaction or, in the case of blockchain, a block (which can contain multiple transactions) is added to the ledger. Digital signatures ensure that the receiver gets the transactions without intermediate parties modifying or forging the contents of transactions, while also ensuring that the transactions originated from senders with access to the private keys.

— **Consensus mechanism**

Once consensus has been reached and the transaction is recorded in the distributed ledger, it is considered final, definitive and immutable.

This requires the implementation of consensus mechanisms – which are the rules and procedures by which consensus is reached.

These rules and procedures that allow the DLT system to maintain and update the distributed ledger and to ensure the trustworthiness of the records in the ledger, i.e. their reliability, authenticity and accuracy - vary widely in implementation approaches.

Example consensus mechanisms include: Nakamoto method with Proof of Work (PoW), Proof of Stake (PoS), Delegated Proof of Stake (DPoS), or Proof of Importance, and Byzantine Agreement Methods including Practical Byzantine Fault Tolerance (PBFT), Federated Byzantine Agreement (FBA), and Byzantine Paxos. Note that hybrid consensus methods employing PoW and PBFT have also been presented (e.g. Zilliqa Consensus).

— **Event distribution**

The event distribution component handles the distribution of events generated within the DLT platform.

— **State management**

The state management component keeps track of the state of assets that are held on the ledger, updating that state when new transactions are committed to the ledger.

— **Secure inter-node communication**

The secure inter-node communication component handles communication between nodes over the network, enabling the operation of the distributed ledger.

The inter-node protocol, also called the backbone protocol runs via this component.

9.3.6 Infrastructure layer

The Infrastructure layer provides the operating environment, including networking, computation and storage components required for the normal operation of a DLT system. It contains the resources including data storage, runtime containers and communication networks. This layer provides the necessary underpinnings for the implementation of the DLT system. It can be provided as a set of cloud computing resources or it can be provided as on-premises equipment.

— **Data Storage**

Physical location to put data for the ledger and other data storage requirements

Data Storage function should meet the following:

— can be deployed and used by each node in P2P network;

— can be deployed distributed or local;

— can support appropriate data sovereignty;

— can provide data writing and query services efficiently, safely and stably.

— **Computation**

Computation function provides the execution capabilities for the operation of the DLT system, including but not limited to container technology, virtual machine technology and cloud computing technology. Generally, runtime environments need to be provided to each node in the DLT system.

— **Communication Networks**

Communication networks are required for the P2P networking of the DLT system nodes and also for communication between the DLT system and the entities in the user layer and in the non-DLT systems.

9.3.7 Cross-layer functions

9.3.7.1 General

Cross-layer functions support the components across all the functional layers. For example, security is needed for the user layer, API layer and DLT Platform layer, therefore security is a cross-layer functional component. Cross-layer functions can support other cross-layer functions as well. In detail, the functions are grouped into development, management and operations, security, and governance and compliance categories.

9.3.7.2 Development

Development functional components support the activities of DLT system developer, including application and system implementation development, build management and test management.

Development functional components consist of

— Interactive development environment (IDE),

— Build management,

— Test management, and

— Secure software development tooling.

To explain these further:

— **Interactive development environment (IDE)**

IDE functional components provide tools for the development of smart contracts, DLT and related applications, including development of support modules. IDE functional components support the use of capabilities provided by DLT platform, including access via APIs, node management and event distribution capabilities. IDEs enable use of functions in the API layer and the DLT platform layer, as well as the infrastructure layer.

The IDE functional component supports the generation of configuration-related metadata for the development of smart contracts; supports the preparation or generation of smart contract configuration scripts and components used by the operators and execution environments on nodes.

— **Build management**

Build management functional components are used to build publishable software packages. The package can be submitted to DLT node owner or operator and deployed in the environment. It can contain the software for smart contract implementations as well as the configuration metadata and configuration scripts.

Build management features include the following:

— support for automated building software packages function;

— provide automated compilation function;

— provide an error message when an error occurs during the build process;

— implement audit during build process;

— build system that provides multi-language support;

— build system that provides multi-platform support.

— **Test management**

Test management functional components support testing of all functions of DLT systems. The components need to generate test reports and provide them with system implementation software to the node owner or operator. Tests are often performed in isolated runtime environments designed to accurately simulate production environments. Without affecting production, test work can also be carried out in a production environment. A test environment needs to be provided by DLT providers, especially DLT system operators.

A fully deployed testing framework needs to include the following capabilities:

— support the management of test plans, test reports, test cases and so on;

— support automatic generation of test reports;

— when test is conducted under the integration of test environment then the production environment need not be affected;

— support testing process automation;

— provide test case library and test database management functions;

— continuous integration/continuous deployment to enable multiple parties request, review, and approve code.

This includes a combination of build management, test management, and governance together to deliver these test capabilities.

— **Secure software development tooling**

The services provided here support the development of secure code by providing automated code review, automated testing and code re-use services. Secure software development tools can include tools to do code review, testing, library, logging and monitoring.

9.3.7.3 Management and operations

Management and operations functional components include a set of functions to manage and control a DLT system to provide its operational capabilities.

Management and operations functional components include

— service directory and catalogue,

— delivery management,

— cross chain service management,

— node management,

— ledger management,

— DLT system management,

— monitoring,

— operations management,

— incident management,

— asset management,

— update, change and version management, and

— business continuity and disaster recovery management.

To explain these further:

— **Service directory and catalogue** - Service directory function provides a list of all DLT capabilities, smart contracts, services and/or APIs of a particular DLT system, provider, or node. The list includes/refers to technical information about deployment, provision and operation of DLT smart contract, service or API. Catalogue functions provide access to information relating to the security architecture and information relating to services, processes and tools. The catalogue can be used to help in the creation and selection of an architecture and its components. Directory and catalogue tools include tools to manage terms, models, product databases and reference libraries.

— **Delivery management** - Delivery management function provides the management function of DLT system delivery, which is provided in the forms of system implementation and access endpoint. At the same time, this function provides necessary workflows to ensure the elements are provided in correct order.

— **Cross chain service management** - Cross chain service management monitors and maintains connections, communication and interactions between two or more different DLT systems.

— **Node management** - Node management function provides management of the DLT platform node implementation including performance and availability usually on one logical or virtual system. DLT platforms can support policies for management, including the ability to use Role Based Access Control (RBAC) to manage creation and manipulation of resources.

— **Ledger management** - Ledger management function provides management of the distributed ledger.

— **DLT system management** - DLT system management function provides management of DLT systems especially for performance and availability.

— **Monitoring** - Monitoring functions include monitoring, analytics, and automation tools that are used to respond to changes in the DLT system and environment. These could include detecting and remediating issues as well as responding to changes in the required system capacity and error analytics.

— **Operations management** - These services include processes to ensure the secure functioning of the DLT system and the recovery of the DLT system from an error state, an incident or problem.

— **Incident management** - Incident management function is a set of processes for detecting, reporting, assessing, responding to, dealing with, and learning from incidents. Incidents and problems can be detected and reported by DLT node or DLT providers or DLT users. Incident management tools include those for incident detection, reporting, analysis, forensics, logging and monitoring.

— **Asset management** - These services achieve and maintain appropriate ownership and oversight of organizational assets. Asset management tools include tools to control asset inventory and track asset ownership.

— **Update, change and version management** - Update, change and version management provide services, processes and tools to ensure that hardware and software are kept up-to-date; upgraded to the latest version; that changes are authorized and managed; and that records of updates, changes and implementations are kept. Tools to support this include change and update management, configuration management databases, software repository, and testing tools.

— **Business continuity and disaster recovery management** - These services assist in preventing, detecting and recovering from an outage, a disruption or an incident.

  In distributed systems, such as DLT, nodes possibly do not require these services; if a node does cease operation, once the node is back in operation, it will automatically receive the latest information. Tools to support this include critical assets and processes to switch over.

9.3.7.4 Security

The security functional components are mainly to provide security attributes such as authentication, authorization, confidentiality, integrity, availability and access control for all the functional layers of the DLT system and the protocols between nodes. These security features are widely used in user and node identity authentication, transaction protocol design, asset protection, communication channel encryption, and application data access control.

Security functional components include the following:

— Asset protection;

— Access management;

— Identity management;

— Cryptography management;

— Assessment and testing management;

— PII protection;

— Availability management.

In a DLT system, the security of information, the protection of the confidentiality, integrity and availability of information and the protection of PII covers more than the ledger; it is a cross-cutting aspect and covers the user, non-DLT systems, API, DLT Platform, infrastructure and other cross-layer functions. As security is a cross-cutting aspect, it supports the other layers in the architecture.

— **Asset protection**

  The services in this area provide logical and physical protection to assets comprising or connecting to some DLT systems. Asset protection services and tools include access control, application security, platform security, network security, and physical security. Some public or permissionless DLT systems might not have the ability to safeguard keys.

— **Access management**

  Access management is relevant to permissioned DLT systems and typically not relevant for permissionless DLT systems. Services can include processes to validate and/or verify the identity of a user before they transact using the ledger or to limit the actions a user can take once successfully authenticated.

  Before making a transaction, a DLT system user typically logs on to a node; the services and processes included in this area would provide the authentication and authorization for that user to successfully log on to the node and then transact. Membership services in the DLT platform layer are used by access management.

  Access management might include identification, authentication, authorization, access control, and access logs.

— **Identity management**

  The services in this area include functions to support the lifecycle of identity. It includes creating DLT accounts and setting attributes and values related to the DLT account, for example, identifiers, roles, and authentication information. It covers entities and roles in a DLT system and can include non-DLT systems connected to the DLT system. Identity management services, processes and tools include role management, digital wallets, and logging.

— **Cryptography management**

  The services in this area include processes to implement cryptography in a DLT system. Tools to support cryptography management include key management and logging.