***Disclaimer***

The present document has been produced and approved by the <long ISGname> (<short ISGname>) ETSI Industry Specification Group (ISG) and represents the views of those members who participated in this ISG.
It does not necessarily represent the views of the entire ETSI membership.

ETSI GR PDL 010 V0.0.5 (2020-11)

**Group REPORT**

PDL Operations in Offline Mode

Release #

<

Reference

DGR/PDL-010\_PDL\_Offline\_Mode

Keywords

PDL, ledger, Smart contracts, Accountability

***ETSI***

650 Route des Lucioles

F-06921 Sophia Antipolis Cedex - FRANCE

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

Siret N° 348 623 562 00017 - NAF 742 C

Association à but non lucratif enregistrée à la

Sous-préfecture de Grasse (06) N° 7803/88

***Important notice***

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

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

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

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

***Copyright Notification***

No part may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm except as authorized by written permission of ETSI.

The content of the PDF version shall not be modified without the written authorization of ETSI.

The copyright and the foregoing restriction extend to reproduction in all media.

© ETSI yyyy.

All rights reserved.

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

Contents

Table of Contents

[Intellectual Property Rights 5](#_Toc66291327)

[Foreword 5](#_Toc66291328)

[Modal verbs terminology 5](#_Toc66291329)

[Executive summary 5](#_Toc66291330)

[Introduction 5](#_Toc66291331)

[1 Scope 6](#_Toc66291332)

[2 References 6](#_Toc66291333)

[2.1 Normative references 6](#_Toc66291334)

[2.2 Informative references 6](#_Toc66291335)

[3 Definition of terms, symbols and abbreviations 6](#_Toc66291336)

[3.1 Terms 6](#_Toc66291337)

[3.2 Symbols 7](#_Toc66291338)

[3.3 Abbreviations 7](#_Toc66291339)

[4 Introduction 8](#_Toc66291340)

[4.1 PDL Overview 8](#_Toc66291341)

[4.2 Reasons For Offline 8](#_Toc66291342)

[4.3 Offline Challenges 9](#_Toc66291343)

[5 High-Level Reference Architecture 10](#_Toc66291344)

[5.1 Introduction to Architecture 10](#_Toc66291345)

[5.2 Type of Nodes 10](#_Toc66291346)

[5.3 Offline Scenarios 10](#_Toc66291347)

[5.4 Spatial Characteristics 11](#_Toc66291348)

[5.4.1 Location of Endorsing Node 11](#_Toc66291349)

[5.4.2 Location of Client Node 11](#_Toc66291350)

[~~4.3.1~~ ~~Recommendations for the Client nodes~~ 11](#_Toc66291351)

[5.5 Temporal Characteristics 11](#_Toc66291352)

[6 Technical Issues Arising From Offline Mode 13](#_Toc66291353)

[5.1 Introduction 13](#_Toc66291354)

[5.2 Offline Client Node 13](#_Toc66291355)

[*5.2.1* *General Considerations* 13](#_Toc66291356)

[*5.1.1* *Ledger Validation* 14](#_Toc66291357)

[*5.2.2* *Storage* 14](#_Toc66291358)

[*5.2.3* *Smart Contract* 14](#_Toc66291359)

[5.2 Offline Validation/Endorsement Node 14](#_Toc66291360)

[*5.2.1* *General Considerations* 14](#_Toc66291361)

[*5.2.2* *Transaction Validation* 15](#_Toc66291362)

[*5.2.3* *Data Storage* 15](#_Toc66291363)

[*5.2.4* *Smart Contracts* 15](#_Toc66291364)

[5.3 Offline Ledger Node 15](#_Toc66291365)

[5.4 Offline Smart Contracts 15](#_Toc66291366)

[5.5 Monitoring Capabilities 16](#_Toc66291367)

[5.5.1 Recommendations for the Client nodes 16](#_Toc66291368)

[6 Proposed Solutions 16](#_Toc66291369)

[6.1 Monitoring Capabilities 17](#_Toc66291370)

[6.2 Validator and Ledger Nodes Architecutral and security considerations and Requirements(can be a chapter) 17](#_Toc66291371)

[7 Offline Architecture Proposition 17](#_Toc66291372)

[8 Offline data storages 18](#_Toc66291373)

[8.1 Introduction: 18](#_Toc66291374)

[8.2 Challenges: 18](#_Toc66291375)

[1) Data Integrity: 18](#_Toc66291376)

[2) Chain merging 18](#_Toc66291377)

[3) Stale transactions 18](#_Toc66291378)

[1 Introduction and Prior Art 19](#_Toc66291379)

[8.3 Introduction To Used Terminologies 19](#_Toc66291380)

[1.1 General Prior Art, Including Public Ledgers 20](#_Toc66291381)

[1.2 Specific PDL Offline Solutions 20](#_Toc66291382)

[1.3 Related Prior Art Enabling Offline PDL 20](#_Toc66291383)

[2 Storage models 20](#_Toc66291384)

[2.1 Offline storage models 20](#_Toc66291385)

[2.2 Online storage models 20](#_Toc66291386)

[3 Protocols and Architectures 21](#_Toc66291387)

[3.1 Offline Mechanisms & Enablers 21](#_Toc66291388)

[3.2 Offline PDL Protocols 21](#_Toc66291389)

[3.3 Offline PDL Architectures 21](#_Toc66291390)

[3.4 Offline PDL Reference Design 21](#_Toc66291391)

[4 Threats and tradeoffs in Offline storages 22](#_Toc66291392)

[4.1 Security threats 22](#_Toc66291393)

[4.2 Potential methods to protect 22](#_Toc66291394)

[4.3 Tradeoffs 22](#_Toc66291395)

[5 Conclusions 22](#_Toc66291396)

[B.1 First clause of the annex 24](#_Toc66291397)

[B.1.1 First subdivided clause of the annex 24](#_Toc66291398)

[History 27](#_Toc66291399)

# Intellectual Property Rights

Essential patents

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

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

Trademarks

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

# Foreword

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

# Modal verbs terminology

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

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

# Executive summary

One paragraph each on:

* general role/importance of ledgers / PDL
* problem statement we are trying to address
* reasons why nodes might be offline
* contribution we make; each subsequent high level section in one paragraph
* summary on high level findings and recommendation

# Introduction

Several paragraphs each on:

* general role/importance of ledgers / PDL
* current work done by ETSI PDL and wide ecosystem
* problem statement we are trying to address
* expanded view on why nodes might be offline
* rational of our work and resulting structure of document

# Scope

The present document describes the current challenges related to data storage and ledger operations when the PDL nodes are offline (duty cycled or truly offline); the methodologies and techniques that can be applied to Smart Contracts to operate when the nodes are offline and develop secure interim storage and negotiation algorithms that ensure the integrity of the data feed to the PDLs.

#  References

## 2.1 Normative references

Normative references are not applicable in the present document.

## 2.2 Informative references

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

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

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

[i.1] ETSI GR PDL001 V1.1.1/(2020-03): "Permissioned Distributed Ledger (PDL); Landscape of Standards and Technologies".

[i.2] European Commission CORDIS: "CORDIS: Projects and Results".

NOTE 1: Available at https://cordis.europa.eu/projects/en

[i.3] ETSI Research and Standards Website /(2020-06-15): "Research and Standards".

NOTE 1: Available at https://www.etsi.org/research

[i.4]

#  Definition of terms, symbols and abbreviations

## Terms

For the purposes of the present document, the [following] terms [given in ... and the following] apply:

**Offline PDL**

## Symbols

For the purposes of the present document, the [following] symbols [given in ... and the following] apply:

## Abbreviations

For the purposes of the present document, the [following] abbreviations [given in ... and the following] apply:

AI Artificial Intelligence

DLT Distributed Ledger Technology

ETSI European Telecommunications Standards Institute

EU European Union

H2020 Horizon 2020

ICT Information and Communication Technology

IoT Internet of Things

IP Intellectual Property

PDL Permissioned Distributed Ledger

# Introduction To PDL Offline Mode

## Introduction

This clause gives a high-level introduction to the Permissioned Distributed Ledger (PDL) in Offline Mode (OFF). To this end, the operational mechanisms of PDL are reviewed first. Then, we illustrate a PDL usage scenario and reasons for PDL going into OFF. Finally, the resulting implications are discussed which lay the foundations for the work carried out in this document.

## PDL Overview

In the most general case, a ledger is a database. A distributed ledger is thus a distributed database that is consensually shared and synchronised across multiple sites, or nodes. The database is typically used through three main interactions:

1. **submit** (content from a client/user onto the ledger);
2. **validate** (and write onto the ledger the submitted content through a consensus protocol); and
3. **read** (the stored content from the ledger).

The content itself has evolved over past years. Initially, only ledger entries could be stored, such as financial transactions, ownership association, etc. However, with the introduction of Ethereum in 2015, execution logic in form of programming code could be stored too. This has led to the emergence of **Smart Contracts** and, as of late, distributed applications (dApps). The role of distributed ledgers is thus evolving from distributed databases, to distributed contracts, to distributed applications.

The **writing of content** onto the ledger typically happens in blocks which are cryptographically linked and, once validated, distributed across the entire ledger fabric. The cryptographic linkage and spatial distribution, along with a properly designed validation protocol, make the data written onto the ledger immutable. The distributed nature of the ledger ensures that no central authority can alter content, thus making this technology useful in the context of non-trusted parties interacting with each other.

At the core of each distributed ledger technology (DLT) is the **consensus** protocol, which is being carried out by the validators. It has many roles but mainly ensures that the same ledger entry cannot appear more than once (thus e.g. preventing the double-spend problem). Different variants have emerged over past years, such as Proof-of-Work (PoW); Proof-of-Stake (PoS), Proof-of-Elapsed-Time (PoET) or Practical Byzantine Fault Tolerance (pBFT). They trade energy efficiency, scale, speed of transactions, among other factors.

Another important aspect is the notion of **public vs private** ledger. It commonly refers to the degree of anonymity of the validators but generally extrapolates to the access rights in general, i.e. who can write to and read from the ledger. In the case of a public ledger, validation and access can be done anonymously and by anybody wishing to participate in the ledger. In the case of a private ledger, validation and/or access is typically handled by a consortium (e.g. 10 companies).

One also distinguishes between **permissionless vs permissioned** ledgers. It defines the degree of trust in the validators which execute the consensus protocol. In a permissionless ledger, anyone can participate in the consensus mechanism; whereas, in a permissioned ledger, only those fulfilling certain requirements can take part in the consensus mechanism. Not all consensus protocols are suitable to all scenarios; e.g. permissionless ledgers (such as Bitcoin) use PoW and permissioned ledgers (such as HyperFabric) use pBFT.

In this document, solely permissioned distributed ledgers (PDLs) are considered.

## PDL Application Example

Figure 1 illustrates an example of a PDL reference use-case scenario, which had been introduced in [REF]. The scenario pertains to an agricultural application, which is explained in more detail below.

One can imagine a farmer of a large set of disaggregated land only usinig natural and organic substances, without any chemical and/or genetically modified substances. To prove these credentials, and thus boosting sales, the farmer decides to join a Bio Certification Alliance. The alliance offers bio certification using a PDL, so as to increase transparency to its alliance governance players, to its farmers and to the end consumer wishing to validate the truthfulness of the bio certificate.

At the farmer’s side, this is enabled through a set of Internet of Things (IoT) sensors measuring chemical and other pollution. These sensors have their trusted certification and unique digital identity. They constitute the **Client Nodes** which transmit information into the PDL for validation. Said validation is done by means of **Validator Nodes**. Once validated, the information is immutably written onto the ledger and stored by means of the **Ledger Nodes**.

In the context of a PDL, the Validator and Ledger Nodes typically belong to a consortium where each member may own a prior agreed set of these nodes. Furthermore, each member or a sub-set of members may offer a set of applications. For instance, a part of the alliance members jointly offers the bio certificate, as long as the sensors in the field support the bio credentials. Another alliance member may offer a smart irrigation service which controls the irrigation system in each of the disaggregated land areas.

Above is enabled through **Smart Contracts** residing in the PDL. Notably, the logic of the smart contract will issue a positive certification flag only when all sensors from each of the fields report adherence to bio credentials. The logic can be programmed to perform that check at regular intervals, or be updated when new data from the sensors in the field arrives. Equally, the irrigation smart contract will trigger the water valves to be opened when moisture falls below a certain level across a prior agreed set of nodes. Other interesting conditions can be baked in, such as only switching on irrigation when the water price is below a certain threshold (unless irrigation is critical to the survival of the crop).



***Fig 1:*** *Example of a PDL reference use-case scenario [REF].*

## Reasons For PDL Going Offline

From above example, it is clear that underpinning technology elements of a distributed ledger can go offline. The reasons of that happening are discussed here:

* **By Design:**
	+ Maintenance: Part of the ledger may require maintenance, such as hardware, networking or software maintenance. Whilst maintenance occurs, the normal operations of the ledger might be impacted which leads to vulnerabilities that need to be dealt with.
	+ Duty Cycle: To minimise energy consumption of the entire ledger, some of its elements might be duty cycled as per a given and prior agreed schedule. Again, whilst nodes are offline, possible operational issues as well as vulnerabilities may occur which ought to be dealt with.
* **By Circumstance:**
	+ Temporary Unavailability: Elements of the ledger may unexpectedly become temporarily unavailable. This could be, for instance, due to an end point moving away from a basestation and thus losing access to the ledger. Another example occurs in very remote locations where the ledger might be connected via a very intermittent fxed-line or satellite network.
	+ Congested Network: In the case of bandwidth-limited or highly congested networks, intra and inter ledger data may not be delivered in time and thus compromise the integrity of the ledger operations. That is, all elements are in principle connected but data is not distributed within the required time limits.
* **By Incident:**
	+ Disaster: Certain elements of the ledger might be powered by power sources which go out of operations, e.g. due to an unforeseen disaster. The sudden disappearance can cause serious issues to the proper operations of the ledger.
	+ Attack: In the case of malicious attacks, elements of the ledger can be compromised and rendered intact or offline; in the worst case, the majority quorum capabilities are compromised. This poses serious threats to the integrity and operations of the ledger.

## Offline Challenges

In the case that a ledger or parts of a ledger are rendered offline, a series of challenges arises which are not normally encountered in fully operational ledgers. These are summarised below and indeed form the rational for the technical approach outlined in the latter part of this document.

From a high level technical and operational point of view, the offline challenges can be summarised as follows:

1. **Security:** In terms of security of the PDL, the following issues arise:
	1. Consensus Capabilities, i.e. the consensus approach underpinning the very essence of the PDL may get compromised.
	2. Weakening Security, i.e. cryptographic primitives may become unavailable and thus certain processes become unverifiable.
2. **Availability:** In the most general terms, availability is impacted as follows:
	1. Reconciliation Time, i.e. when some nodes come back online after certain offline time, it may take some time for them to catch up with the ledger. In the case of mobile nodes, it may be possible that by the time they reconcile, the service is interrupted again and they go offline again.
	2. Side-Chains & Chain Merging, i.e. the offline mode will trigger the emergence of side chains which has an impact on integrity and availability; and certainly will influence the approach to chain mering. The latter is typically a rare event but may happen very frequently in the discussed scenarios of offline operations.
	3. Stale Transactions, i.e. if chain merging is not successful, some transactions may become stale which ought to be dealt with by the offline ledger design.
3. **Integrity:** Important issues arise in the context of data integrity:
	1. Control Data, i.e. data a previously offline node need to have in order to be accepted back into the PDL network. This typically pertains to authentication data, cryptographical keys, connectivity data, etc. The data related with the content of the ledger itself is out of scope here.
	2. Software Code, i.e. before being accepted back into the PDL network, checks will need to be performed to ensure that the software/program running on it have not been compromised.

# PDL Offline Scenarios

## Introduction

This clause discusses possible scenarios arising as a result of the reasons ledgers going offline, as discussed in clause 4. To aid technical understanding, we first introduce a high-level reference technical architecture. Thereupon, we introduce the role of the different types of nodes. This allows us to construct and discuss the different PDL offline scenarios. Last, we discuss temporal and spatial characteristics emerging from the different offline scenarios which is unique to a permissioned ledger having offline constituents.

## High-Level Node Reference Architecture

The application example given in clause 4.3 can be abstracted to a high-level reference ledger node architecture as illustrated in figure 2.

Notably, a set of client nodes collects data which ought to be written onto the PDL. The data from the client nodes is transmitted via a fixed or wireless network to the validator nodes.

The validator nodes prepare the data for the specific ledger, i.e. sort the data, cast it into a specific format, check for initial consistency, etc. They then perform the PDL-specific consensus protocol to validate the content of the information provided by the client nodes.

Once validated, the content is written onto the ledger nodes which store the content for perpetuity. Ledger and validator nodes belong to several parties.

The end-to-end fabric is configured and maintained by means of PDL-orchestration. Said orchestrator could e.g. sit on one or several governance nodes.

The links between the client and validator nodes and between validator and ledger nodes can be volatile due to the presence of wireless networks or highly congested fixed networks.



***Fig 2:*** *High-level node reference architecture comprised of client, validator and ledger nodes; and overseen by orchestration / governance capabilities.*

## Type of Nodes

As discussed in clause 4.4, there can be several reasons for a node to go offline, such as an engineered duty cycle or an unexpected outage of the wireless network. In this clause, we explain the role of each node type and estimate the likelihood of them going offline.

* **Client Nodes:** These are nodes which belong to a client collecting, storing and/or transmitting data. From the ledger’s perspective, these are temporary and transient nodes: temporary because a specific client node may lose connection with the ledger (e.g. a car driving is going through a patch of poor connectivity) and transient because clients may sign onto the specific PDL service and after a few months/years sign off. Since their presence cannot be guaranteed, they do not take part in the consensus process nor do they form part of the storage ledger. They typically only update their state by sending transactions to validator nodes.
* **Validator Nodes:** These nodes accept transactions from the client nodes, check the validaty and send them to ledger nodes. Whilst validator nodes are typically hosted in operationally reliable locations, they can go offline when e.g. there is a network congestion or poor mid/backhaul. For that reason, a strong governance model is required which ensures the viability of the consensus approach. For example, a requirement could be that consensus can only be reached if two-thirds of the validator nodes are online.
* **Ledger Nodes:** Ledger nodes are permenant nodes of the ledger. This means that all of the nodes are generally available and online, unless of course in the case of force majeur or a cybersecurity attack. Depending on the design requirements these nodes can or cannot be validator nodes; indeed, some of the architecture models may merge ledger nodes and validator nodes. The state of the ledger nodes is being updated by the validator nodes.
* **PDL-Governance/Orchestration:** Governance is important as it ensures the proper monitoring and execution of the PDL. It is particularly important in the offline scenario as the orchestration capabilities will require to maintain viability of the consensus protocol and ledger operations, among others. It may include roles like the verification of certificates or the revocation of access rights.

Each of these nodes require trusted digital identities and trusted certificates. Operationally, they can be hosted on baremetal servers, virtual machines (VMs) or containers (managed by e.g. Kubernetes).

## Offline Scenarios

The possible scenarios arising due a subset of certain node types becoming unavailable are summarised in Table 1. Note that when nodes are offline, they may or may not be functional. If they are functional, then they can continue executing tasks locally. Note further that OFF refers to at least one of the node types not being reachable, and not necessarily all of them. The table is sorted from the most likely scenario to the least likely one.

* **Scenario #1**: It is the reference scenario where all nodes are functional and reachable. This should be the modus operandi of the PDL.
* **Scenario #2**: Occasionally, one, several or a cluster of the client nodes may go offline due to reasons discussed in previous clauses. When this occurs, in most cases one would assume the nodes to be functional, just not able to reach the PDL. Situations where the nodes have been rendered infunctional or where an entire cluster of nodes has gone offline should also be catered for.
* **Scenarios #3 & #4**: Rarely, the or a set of validator nodes goes offline for reasons discussed in previous clauses. Again, one needs to cater for situations where the node has been rendered infunctional, and not just temporarily gone offline.
* **Scenarios #5-#8**: Extremely unlikely but plausibly, one or a set of ledger nodes goes offline for reasons discussed in previous clauses.

The likelihood of any of the scenarios to occur will depend on the spatial distribution of the nodes, which also impacts the temporal behviour of the system. Unlike always-on PDLs, Offline PDLs thus suffer from breaks in chain causality. This, in turn, jeopardises the very essence of distributed ledgers and thus warrants special design attention.

***Table 1:*** *Possible operating scenarios due to different node types being reachable or offline. Note that when offline that they may or may not be functional. Note further that OFF refers to at least one of the node types not being reachable. The table is sorted from most likely to least likely.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Client Nodes** | **Validation Nodes** | **Ledger Nodes** | *Likelihood* |
| **Scenario #1** | ON | ON | ON | *very likely* |
| **Scenario #2** | OFF | ON | ON | *occasional*  |
| **Scenario #3** | ON | OFF | ON | *rare* |
| **Scenario #4** | OFF | OFF | ON | *rare* |
| **Scenario #5** | ON | ON | OFF | *unlikely*  |
| **Scenario #6** | OFF | ON | OFF | *unlikely* |
| **Scenario #7** | ON | OFF | OFF | *unlikely* |
| **Scenario #8** | OFF | OFF | OFF | *unlikely* |

## Spatial & Operational Characteristics

The likelihood of a node or set of nodes going offline depends on spatial and operational characteristics of the system, i.e. the node location and operational provisioning.

The spatial composition of a typical PDL is likely as follows:

* **Client Nodes**: These are typically lightweight nodes with an embedded OS and limited processing power, onboard memory and battery power. Depending on the PDL design, they could form part of the PDL fabric or not. In any case, client nodes write data onto the ledger and may receive instructions from the ledger via smart contracts. They are located at the very edge of the network, are mobile and often untethered.
* **Validator Nodes:** These are typically placed physically close to the client nodes but within the networking infrastructure. For instance, in a 5G system, validator nodes could sit in the CU of the RAN, or any other edge-cloud location. They would typically be virtualised via VMs or containers, and have sufficient processing power. Whilst spatially placed in a well-managed environment, the edge is often connected via a volatile backhaul which makes the operational viability of these nodes volatile.
* **Ledger Nodes:** These are typically placed in public or private data centres. They are well powered and well networked. They have sufficient processing power and storage capacity. They are also typically virtualised via VMs or containers.

To minimise operational outage, all nodes in the system should be provisioned as follows:

* Physically Protection: Ideally, each node ought to be physically tamper resistant, whether IoT Client Node or Validating/Ledger Nodes. This will ensure that the certificates stored on the nodes cannot be compromised.
* Power Provisioning: Each node ought to have one primary and ideally a secondary backup power source so that no power outages occur during the operations of the ledger.
* Cyber Security: Each node should be properly protected against cyber security attacks, DDoS attacks and other means of getting access to the nodes.
* Networking Connection: Each node ought to be properly networked, using networks which are reliable without outages nor congestion.
* Critical Network Path: The PDL ought to be designed such that there is no static and single network path dependency which is critical to a plurality of nodes. In other words, enough network diversity ought to be provided so that when one network path fails, another can be used to connect the node.
* Monitoring Capabilities: The PDL ought to have network, storage and compute monitoring capabilities so that the orchestrator obtains reliable real-time information on the state of the PDL.

Whilst above is an ideal operational scenario which largely holds true for Ledger Nodes, Validator Nodes which reside at the edge of the network will likely have the following differences in a real-world deployment:

* Networking Connection: Whilst each node is connected, an edge may experience a poor backhaul network and thus be not reachable over a givern period of time.
* Critical Network Path: Also, an edge may only be connected via a single network path (e.g. a single fiber link or single satellite link) thus not providing enough diversity in case that link goes down.
* Monitoring Capabilities: Whilst monitoring capabilities might be in place, above-discussed networking problems may render the monitoring, and thus orchestration, capability ineffective.

Client Nodes (e.g. IoT sensors or actuators) will be even less equipped to operate in ideal operating conditions:

* Physically Protection: Client nodes in the field are often exposed and without supervision, thus making it difficult to guarantee physical protection and tamper resistance consistently.
* Power Provisioning: Client nodes in the field also may experience power supply issues due to, e.g., batteries running low, solar panels not producing enough power, etc.
* Cyber Security: Client nodes in the field are also exposed to cyber attacks, mainly because embedded systems are not security patched at the same level as traditional IT equipment.
* Networking Connection: For nodes in the field, networking may be volatile over a short period due to wireless network congestion or fading/shadowing blockage. The outage may also be seasonal due to, e.g., trees growing leaves back and thus obstructing an already weak wireless link.
* Critical Network Path: Client nodes rarely have networking diversity, i.e. they are typically connected via one networking interface only. Therefore, if that link breaks, the node is not accessible.
* Monitoring Capabilities: Due to the embedded nature of the devices, monitoring capabilities will be limited. Furthermore, the duty cycling of the devices and the intermittent networking conditions will prevent real-time monitoring all the time.

## Temporal Characteristics

An important element of Offline PDLs are the temporal characteristics of the involved nodes, where we differentiate between the following levels of planned and unplanned intermittence:

* **Schedule:** In this case the node is scheduled on purpose by the owner(s). This could e.g. be done in the context of duty cycling to save energy. Whilst the schedule may have a limited impact onto the node itself, it may have a strong impact on any dependent smart contract. For instance, a smart contract may only work when a certain amount of nodes provides information within a given time window which will require the schedules of the nodes to be synchronised. Clearly, there is a strong governance issue when it comes to schedules, which – in the context of PDLs – goes beyond simple energy savings.
* **Graceful Disappearance:** Client nodes or even validating nodes can disappear gracefully in that the connection becomes weaker and weaker over time. In the case of a validating node, this could be because the network has peaks of congestion which increase response times until the outage occurs. Equally, client nodes may suffer from network congestion or simply a weakning wireless connection as the client node moves away from the basestation/access point. In principle, the PDL system could learn to recognise such events and initiate contingency protocols before the connection is cut off. For instance, in the case of the validating node, another or a prior-established proxy node could be made validating node. In the case of a client node, this may give an opportunity to establish a trusted local environment which is able to operate offline for a given period of time.
* **Sudden Disappearance:** Nodes may of course disappear suddenly, without any prior warnings or indications. The reasons could be manifold and range from non-malicious due to power cut, equipment failure, or sudden network blockage; to malicious due to a DDoS attack or other cybersecurity breach.

In the context of temporal chacteristics, it is important to pay attention to the **causalilty of the events**. For instance, the order in which some client nodes or validating nodes become unavailable is important to the proper functioning of the PDL and smart contracts. It will impact recovery/auto-recovery protocols, the way majority quorum is done, the way how governance is being executed, among others.

# Technical Issues Arising From Offline Mode

## Introduction

This clause discusses the technical issues which arise from elements of the PDL going offline. To this end, we discuss the technical issues arising from the points of view of a client, validator and ledger node. Furthermore, we discuss specific issues arising in the context of smart contracts, as well as monitoring and cybersecurity approaches.

## Offline Client Node(s)

## General Considerations

In the context of client nodes, technical issues arise in the following situations:

* **Client Data:** A major concern here is how to secure and ensure a trusted environment for the client node when in offline mode.
* **Ledger Data:** Client nodes may need access to data which resides on the ledger which is not possible when in offline mode.
* **Smart Contracts:** Smart contracts need to remain operational, in the case when client nodes go offline.
* **Chain Reconciliation:** A major challenge is to enable the reconciliation with the main chain, once the offline node gets connected again.

These issues are now discussed in greater detail in subsequent clauses.

## Securing The Offline Data

## Enable Access To Ledger Data

## Enable Smart Contract Operations

## Enable Chain Reconsiliation

## Offline Validator Node(s)

This clause discusses the technical issues which arise from elements of the PDL going offline.

## Offline Ledger Node(s)

This clause discusses the technical issues which arise from elements of the PDL going offline.

## Offline Smart Contract

This clause discusses the technical issues which arise from elements of the PDL going offline.

## Monitoring Capabilities

This clause discusses the technical issues which arise from elements of the PDL going offline.

**Privacy of monitoring as maybe some consortium members do not want to share information about the state of their part of the system.**

## End-to-End Cybersecurity

This clause discusses the technical issues which arise from elements of the PDL going offline.

# Technical Issues Arising From Offline Mode

## Introduction

…

## Offline Client Node

## *General Considerations*

* **CONCERNED ABOUT:**
	+ Secure the offline data
	+ Enable access to ledger data
	+ Enable smart contract operations
	+ Enable reconciliation with the main chain
* Also consider proxy nodes for smart contract scenario (scenario where node may not want to advertise it’s conditions of meeting a smart contract trigger; for instance hiding from competitors --- TBC how this is practically done)
* Make sure that offline generated content cannot be compromisesd:
	+ Secure vault to temporarily store data / blocks
	+ Consider using time-limited assymetric key derived from the moment the PDL was last accessible
	+ Consider building a local PDL through Kubernetes clusters (tbc)
	+ Ramp up local PDL which could be time limited eg deleted after a certain time to save memory (Tbc)
* Think on how to reconcile content on main PDL:
	+ Keep OOB low bandwidth link to main ledger?
	+ Constrained devices result in nodes that do not maintain the ledger, i.e. replicate the ledger. These nodes do, however, communicate with the ledger through an API; hence, offline services integrated into the API might provide better speed and scalability options.
		- The API should have the necessary facilities/functions for creating the time-limited asymmetric key.
		- The API would then also be responsible for ensuring that the data to be uploaded is in the right format, and has the right credentials.
	+ Insert into PDLwhen online again
	+ Post-insert and re-run consensus (hindsight consensus)
	+ Post-insert but run “zipper” consensus principle
* The main aim of having a set of “orderers” in consensus for permissioned ledgers is to increase the rate of finality which is predicated on reducing the frequency of forks, or removing the possibility of forks. Hindsight consensus would in-effect create forks in the ledger and ultimately disrupt the deterministic execution of transactions.
* Hindsight consensus would also increase the energy consumed in consensus but not to the magnitude required to re-validate the previously accepted blocks, from the block height of the hindsight consensus.
* For nodes to create an offline ledger (side chain), the nodes would need to be able to create blocks while offline. If a consensus model such as the one seen in Hyperledger is assumed, each node in an offline state would assume the roles of “endorser, orderer, and validator”, to successfully append blocks to the offline ledger.



Figure 1 The view of the internal ledger of a node that has previoulsy been offline. The yellow blocks indicate the entries of the node to the ledger while disconnected from the p2p network.

## *Ledger Validation*

## *Storage*

## *Smart Contract*

## Offline Validation/Endorsement Node

## *General Considerations*

* Today unlikely to disappear but think future where we may have these nodes at the edge;
* there might be problems with the special security certificates, e.g. they may expire
* Consent protocol (CFT – Crash .. ) should tolerate this but we need to see causality issues with smart contracts
* These nodes may make access to 3rd party storage/etc infrastructure, such as 3rd party databases (eg Oracle™ SQL/NoSQL databases specifically geared towards DLT); we should see how this impacts offline operations [for Chonggang/Tooba to revise; Trustworthy Blockchain Oracles: Review, Comparison, and Open Research Challenges (https://ieeexplore.ieee.org/abstract/document/9086815)]
*According to this article (similar definitons in other articles too): "Despite the decentralized and trustless architectures of the blockchain systems, smart contracts on their own cannot access data from the external world. Instead, smart contracts interact with off-chain external data sources, called oracles, whose primary job is to collect and provide data feeds and input to smart contracts. "*
* **CONCERNED ABOUT:**
	+ Graceful disappearance eg due to network congestion (assign proxy node?)
	+ Sudden disappearance, thus disrupting ongoing session (MPTCP?)
	+ Disappearance of a critical mass of nodes, thus violating quora conditions

Consensus nodes are the type of nodes which partipate in the offline consensus, that is to say, that when the some of the nodes are offline, they can still run consensus among themselves. These nodes can be called as “backup nodes”

There are several situations, when the backup nodes that is the nodes those are running offline consensus, take over and run their own consensus which depends on the service providers and governance of the PDL. Some of them can be as follows:

1. When the signal quality reached below a threshold
2. When some number of base stations are offline (possibly due to malicious activity)

## *Transaction Validation*

## *Data Storage*

## *Smart Contracts*

## Offline Ledger Node

* **CONCERNED ABOUT:**
	+ Unless reaching critical amount of nodes, this is dealt with by the consensus protocol

Normally dealt with through consensus protocol

* Also explore proxy node assignment when gradual decrease in connection can be quantified
	+ How can link deterioration trigger a proxy request? (Should the offline node ID be transferred to the proxy node? Important from a smart contract; should there be a specific trigger to a specific other consensus node; or should be a general request which then chooses a random consensus node; so we need to check also the “unproxy” mechanisms to avoid malicious attacks.
	+ Consensus nodes in PDL are typically fixed per player (typically 1 consensus node per organisation; if there are more then either fair balance or a specific reason for an org to have more); Consider internal transfer of concensus node rights (deterministic vs random; what happens if there is no replacement within the organisation; etc)
* How do we account for false positives. For example, a temporary interference in signal of 3-5 seconds might cause the consensus node to request a proxy, but then might require the voting rights back immediately after.
* Another example might be a high retransmission rate which occurs during network peak periods. This might create a scenario where a proxy is necessary for he duration of the peak period.

## Offline Smart Contracts

*We need to see if this warrants a separate section of whether this shouldn’t be absorbed in above sections.*

* Consider reverse problem in that smart contract needs to be downloaded locally
* A scenario such as where a cluster of devices depend on the smart contract for the finality of certain operations but have lost connection to the ledger network. This loss of connection might be due to the more resource-heavy node going offline.
* A successful offline smart contract system would require
	+ A trusted environment where the virtual machine (for example EVM) required to run the smart contract is hosted.
	+ The bytecode for the specific smart contract must be located on the node and a hash proof of the validity of the smart contract, i.e. the smart contract on the node is the same as that on the ledger.
	+ The resulting output from the execution of the offline smart contract must be synchronised to with the state of the online ledger. This synchronisation cannot happen individually to avoid a replay attack/scenario. Therefore, the synchronisation might involve a pointer to the block containing said transactions or be a singular entry of the transactions between two nodes, on the online contract.
* The obvious challenge would be reconciling the offline states with the online states. For example, the current price bid for oil online vs the submitted price bid for oil offline.
* How to create secure and trusted environment locally; how to execute locally; and how to inform back the execution results/confirmation (see Section 4.2).

## Monitoring Capabilities

* Duty cycling (orchestrator for duty cycles to ensure that cycles are reconciled)
* Graceful deterioration (proxy nodes)
* Sudden disappearance (alerts)
* Monitoring/probing infrastructure (cloud, Kubernetes, independent PDL)
* Monitoring nodes integrity whilst they are offline? Other methods then PDL – can be traditional ones.

### Recommendations for the Client nodes

*This can include:*

*How they will reconcile with the ledger*

*Should they keep the full ledger or light-weight*

## End to end Cyber Security

# Proposed Technical Solutions

## Introduction

* Explain that this section is dedicated to technical solutions which are being proposed as a result of the issues identified in Section 6
* Walk through each subsection from a high level point of view and explain how they relate; explain that we take a holistic view here

## Ledger Monitoring Capabilities

* Mutual check if other / important nodes are alive
* Observe gradual and/or sudden degradation of links / nodes
* Direct monitoring (ask if node is alive) vs continuous indirect monitoring (measure eg RTT of ACKs)
* Monitor transaction / validation speed
* TBC: Monitor memory allocation/availability (situations when the nodes are virtualised/VM/K8s)

The status of PDL nodes can be monitored in different ways. The monitored PDL node status (e.g., online or offline) can be maintained at an orchestration node and can be leveraged to determine a proxy node for a particular PDL node when the PDL node goes offline. Some example approaches for monitoring PDL node status are described below:

* PDL nodes can mutually check each other’s status. For instance, a PDL node A can actively send a request to its neigboring PDL nodes (e.g., B and C) to check their current status. After that, the PDL node A can report the status of itself and its neighboring PDL nodes to the orchestration node.
* A PDL node A itself may obserse the gradually degraded quality of its communication links to other PDL nodes. As a result, the PDL node A may actively send a status report message to the orchestration node and/or its neighboring nodes, before it loses connectivity.
* The orchestration node may directly solict the status report from a PDL node, which then sends a status report message to the orchestration node.
* A PDL node A may continuously and indirectly monitor the status of its neighboring PDL nodes (e.g., B and C), for example, by measuring the number of transactions from neighboring PDL nodes, or by measuring the round-trip-time (RTT) between itself and neighboring nodes. As an example, if the PDL node A suddenly stops receiving any transaction from PDL node B but there are still some transactions coming from PDL node C, the PDL node A can record PDL node B’s current status as offline. Then, the PDL node A can report PDL node B’s status to the orchestration node and/or other PDL nodes.

## Node Proxy Mechanisms

* Based on gradual degradation of links, a validating node could eg nominate a proxy node to carry out the validation once the node itself goes offline
* Pool of Proxy Nodes could be selected/established before its being nominated, thereby ensuring that the proxy nodes are secure/authenticated
* mechanism which allocates proxy nodes out of the Pool of Proxy Nodes (should follow the same principles as the overarching PDL)

## Secure Offline Operation – Nodes View

* *Solution(Eusebio):* Trusted Execution Enviornment ) TEE ---
	+ TEE should be installed at the client’s end to maintain client/chain data integrity – Eusebio will contribute as section or a paragraph.
	+ TEE should be done by default when ramping up the client node or when offline is about to or has already occurred
* Ramp a local PDL within the offline environment – check that there is critical mass in terms of number of nodes or number independent trusted environments
* Propose solution for smart contracts: maybe pre-emptively download pertinent smart contracts to edge nodes in case LAN-PDL needs to be ramped up? Main security problems:
	+ Security at the offline edge
	+ Monitoring entity at main ledger which ensures that nodes which come online again are authentic
	+ *Solution: see dynamic, time limited keys*
* Need to see how “interim” the interim storage; need to write out the exact operation of the interim storage whilst offline (how/when established; how operated; how/when terminated)
* What needs to be done immeditaly when node comes back online?

## Secure Offline Operation – Ledger View

* What happens if an online node triggers a smart contract in the PDL which requires an offline node (eg the data, some action, or anything else) --- what should be done then?
* Could we have Conditional Smart Contracts? Or Timedelayed Smart Contracts?
	+ Could be infinitely complex and lead to non-causal scenarios which cannot be solved
	+ Change smart contracts such that they trigger (again, or repeatedly) until execution is guaranteed 🡪 create a form of memory in the smart contract
* Once offline node comes online, it may trigger a smart contract
	+ How to do this exactly
	+ We need to check the temporal/causal characteristics (some conditions are met when nodes are offline but not met when all nodes come back online)
	+ We may need to specify this in the PDL smart contracts
	+ The node which comes back online should be responsible for reconciling the offline information; could be done with the help of the PDL orchestration layer
	+ Ensure there is no infinity loop
* Validating nodes going offline (eg a base station cannot be reached because of mid/backhaul networking issues):
	+ Commence the Proxy Node Pool protocol, ie make sure that a proxy is chosen and made operational

## Reconciling Offline – Online Ledgers

* What exact procedures need to be done to reconcile data, control data (security, etc) and smart contracts

## ~~Validator and Ledger Nodes Architecutral and security considerations and Requirements(can be a chapter)~~

Proposed solutions

3 – Attack – if some attacker brings down all or majority of the validating nodes – Chongong – is there a way to identify the vulnerable nodes (vulnerable to attacks), it may be possible to bring down few nodes for sometime and run safety/security check periodically before bringing them back. Eusebio – Another aspect of possible attack, circulation of power of validity - It is also possible through (Trusted Execution Enviornment ) TEE, chaincode/smart contract might have some authority to change the role of node.

# Offline Architecture Proposition

Glues together prior PDL work, as well as the proposed solutions in the context offline operations.



**do example for telco (where in CU/DU; where in CP/UP)**

+++++ FROM HERE ONNWARD NOT TO BE DISCUSSED DURING OFFICIAL PDL CALL ++++

SEGMENT FROM TOOBA (15 Dec 2020)

# Offline data storages

## Introduction:

PDLs are distributed data storages, which means that every node in the network keeps a copy of the ledger; when an update comes all the nodes synchorize their local ledgers accordingly. The problem comes when some of the nodes are offline and they want to update the ledger; if a ledger is unreachable to nodes updating the ledger state, correct information cannot be available to the ledger. This is particularly very important when a transaction is dependent on other transactions. For example, if transaction A is dependent on transaction B, and the node sending sending transaction B is offline, and some other node X and sends transaction C and changes the state of transaction A. A mechanism to storage the data is required, to maintain the data integrity. Two major scenarios when a ledger or some of the nodes of ledgers are offline are discussed below:

1. One of the node is offline: This is a simplest scenario, when one of the node is offline. In this case, the node may have to store the data at some iterim storage to prove the integrity of the data.
2. Some of the nodes are offline: If two or more of the nodes are offline, they can create their own side-chain and fork to the main ledger when they go back online

## Challenges:

The major problem here is that, the state of PDLs changes very quickly and by the time offline nodes come back online, the main ledger might be on a completely different state.

### Data Integrity:

One of the major problems is the data integrity, PDLs achieve integrity through ledger replication to all of the node. When a node is offline when the data is stored at some iterim/middle storage, the main problem is the storage features this particular storage uses. Other mechanism of storage such as offline node establish a side-chain, can be adopted and discussed in clause 5.

### Chain merging

One possible solution, is that all the offline nodes establish a side-chain, and when they come back online, update the ledger. Since, many nodes have saved the data, and through side chain mechanism established conesus among themselves, can prove integrity of the data to the main legder. However, there are two main problems:

1. The main chain state can be moved forward and the old transactions (that is transactions in offline mode) are no longer applicable
2. If there are number of side-chains merging to the main chain/ledger number of factors should be considered such as latency of the merging(TBD).

### Stale transactions

As discussed in the earlier clause, the transactions in offline mode can become invalid after certain time; particularly when the target state is updated by some other transaction (from an online node). This can cause of monetary losses particularly; for example, if A and B are bidding for item I, and A bids higher then B but goes offline before the bid is received by the ledger, then item I may be sold to B.

When a PDL is offline, one or many nodes cannot access the main ledger, this means, records cannot be updated to the main ledger. The problem is when the offline nodes come back online, they need to prove the integrity of the data to nodes of the main ledger. Some of the methods to prove the integrity of offline ledgers is as follows:

1. Maintain a side-chain : The offline nodes can establish connection between themselves and start a side-chain, and when the main ledger comes back online they can fork their branch to the main ledger. In this strategy there are two problems, 1) there should be a minimum number of nodes that establish a side chain. Second, when number of nodes are offline and many side chains are created, forking many of them to the main chain can be a problem.
2. Cryptographic Means: In this strategy, the nodes can secure the data with pre-agreed keys(TBD)

# Introduction and Prior Art

\* discuss structure of this section

The necessity for the classification of an offline mode stems from the need to preserve and validate data that was generated when a previously online node loses connection with the rest of the p2p network. While offline full nodes are able to perform operations based on the most up-to-date version of the local ledger, the rest of the Blockchain network does not benefit from the result of the operations. These results must be cryptographically secure and timestamped in a manner that ensures the successful appending of the results to the ledger, when connection is resumed. Hence, a secure model that promotes offline security of data and the protocol for arriving at successful consensus on the back-dataed data is necessary.

From a security perspective, an offline mode might be desireable as it limits the exposure of the node to attacks i.e. less opportunities for remote attacks and thefts.

## Introduction To Used Terminologies

**Real-Time PDL**

A PDL node is considered to be in a “real-time” stater when two conditions are met:

1. The node does not have to initiate a “pull” operation i.e. the pushed blocks from other nodes sufficiently synchronise the node to the global state.
2. All nodes in the node list are “active” or “alive”. This provides the nodes with the confidence that the local ledger is up-to-date and that its pushed transactions are disseminated in real time.

Other metrics may be used such as the number of blocks needed the synchronise the node to the global state must not exceed a set threshold, as this may indicate that the node has been considerably offline. An example of such capping or threshold could be the number of blocks or block headers received in the last round from peers are 10 or less. This value can be substituted for a more adaptive figure such as 1-third of the max downloadable block/header count per message.

**Synchronous PDL**

**Asyncrhonous PDL**

A node can be considered to be in an asynchronous state when its synchronisation is periodic or where synchronisations are a function of a "pull" request. Traditionally, a node in an "active" state indicates that the node is available for push notifications. This node may still sign “alive” messages to other p2p nodes in the network. However, an additional parameter could be set that identifies the synchronisation preference of active nodes. This parameter would determine the synchronous or asynchronous state of a PDL node.

**Offline PDL**

Simply put, an offline node is a node that has no active connection to any synchronous or real-time PDL node. This node might be offline due to being duty-cycled of an inconsistent internet connection. This node might still carryout PDL related operations outside of the network either based on data from its most recent view of the ledger or self-generated data. Regardless, this data must be preserved in a state that it can be validated and appended upon connection.

## General Prior Art, Including Public Ledgers

Hardware-based Implementations

A cybersecurity company in Israel, GK8, successfully deployed an offline blockchain node. Similar to the offline node proposed by Cryptofuse, this node is fully offline and requires a tailor-made hardware to ensure certain security considerations. Their models are built around the node being “fully offline” i.e. the node will never have an active connection to the p2p network. These nodes through a combination of processes successfully pubish to the ledger network, however, all input or updates from the p2p network are confronted by several firewalls before being appended to the local ledger.

## Specific PDL Offline Solutions

## Related Prior Art Enabling Offline PDL

# Storage models

## Offline storage models

## Online storage models

# Protocols and Architectures

## Offline Mechanisms & Enablers

Everything about underlying tech enablers

##  Offline PDL Protocols

Protcols resulting from above

A protocol similar to the GHOST protocol adopted in Ethereum might serve useful for appending back-dated transactions into the ledger. When the node is offline, all PDL operations and storage should be regarded as a side-chain operation. While these operations remain questionable to the rest of the network, the credentials of the side chain to the local node is valid. Hence, the transactions and storage on the side chain are valid, pending when there is a peg that either reference the data stored or facilitates the cryptographic operations necessary for appending the data to the chain. A peg refers to the protocol involved in achieving consensus on the data stored on the local side chain. The figure below demonstates the local storage of offline transactions and the continual synchronisation of the global ledger, while maintaining the local ledger. The protocol has to ensure the global chain will not be broken due to the presence of the offline, side-chain.



## Offline PDL Architectures

Architectures resulting from above

While offline data may be pushed onto the ledger, it should never be used to enhance or ensure any security protocols.

## Offline PDL Reference Design

TBC but would be good to extract from above a single reference design; that could help with future TS documentation.

# Threats and tradeoffs in Offline storages

## Security threats

## Potential methods to protect

## Tradeoffs

# Conclusions

Annex A:
Title of annex

Annex B:
Title of annex

# B.1 First clause of the annex

## B.1.1 First subdivided clause of the annex

Annex:
Bibliography

Annex :
Change History

| Date | Version | Information about changes |
| --- | --- | --- |
| <Month year> | <#> | <Changes made are listed in this cell> |
|  |  |  |
|  |  |  |
|  |  |  |

# History

|  |
| --- |
| **Document history** |
| v0.0.1 | 12 Nov 2020 | Creation of document, TOC and high-level content |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

*Latest changes made on 2019-09-10*