***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 - v0.0.3 (2020-01)

**Group Report**

Smart Contracts in Telco Permissioned Distributed Ledgers — System Architecture and Functional Specification

Reference

DGR/PDL-004

Keywords

blockchain, Smart Contracts, Policy, Service Level Agreement (SLA)

***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](http://www.etsi.org/standards-search#Pre-defined Collections)

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 only prevailing document is the print of the Portable Document Format (PDF) version kept on a specific network drive within ETSI Secretariat.

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 2020.

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

Intellectual Property Rights 4

Foreword 4

Modal verbs terminology 4

Executive summary 4

Introduction 5

1 Scope 5

2 References 5

2.1 Normative references 5

2.2 Informative references 5

3 Definition of terms, symbols and abbreviations 5

3.1 Terms 5

3.2 Symbols 5

3.3 Abbreviations 5

4 Smart Contracts 6

4.2 Definition 6

4.3 Smart Contract Programming Paradigms 6

4.3.1 Object-Oriented Paradigm 7

4.3.2 Properties 7

4.3.2.1 Immutability 7

4.3.2.2 Availability 7

4.3.2.3 Transparency 7

4.3.2.4 Self-Execution 7

4.3.2.5 Reusability 8

4.4 Components 8

4.5 Storage 8

4.6 Life cycle of a Smart Contract 8

5 Smart Contracts – Planning, coding and Testing 8

5.1 Planning phase 8

5.1.1 Verification 8

5.1.2 Validation 9

5.2 Coding Phase 9

5.3 Testing Phase 9

5.4 Modelling 9

6 Architectural requirements for Smart Contracts 9

6.2 Reference Architecture 9

6.3 Interaction between PDLs 9

6.3.1 Membership Service Provider (MSP) 10

6.3.2 Smart Contracts (SC) 10

6.3.3 Node Identity 10

6.4 Inter-PDL communication 10

7 Smart Contracts – Applications, solutions and Needs 10

7.2 Regulatory Aspects 10

7.3 Security of the Contracts 10

7.4 Enforceability 11

7.4.1 The role of the regulatory authority: 11

8 Limitations of Smart Contracts 11

8.2 Inter and Intra system threats 11

8.3 Limitations 11

Annex B: Title of annex 12

B.1 First clause of the annex 12

B.1.1 First subdivided clause of the annex 12

Annex <L>: Authors & contributors 12

Annex <L+1>: Bibliography *(style H9)* 12

#

# 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](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 maybe, 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

*[ETSI Drafting Rules](https://portal.etsi.org/Services/editHelp%21/Howtostart/ETSIDraftingRules.aspx)* [(](https://portal.etsi.org/Services/editHelp%21/Howtostart/ETSIDraftingRules.aspx)*[EDRs)](https://portal.etsi.org/Services/editHelp%21/Howtostart/ETSIDraftingRules.aspx),*

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

The present document is part <i> of a multi-part deliverable. Full details of the entire series can be found in part [x] [Bookmark reference].

# Modal verbs terminology

*[ETSI Drafting Rules](https://portal.etsi.org/Services/editHelp%21/Howtostart/ETSIDraftingRules.aspx)* [(](https://portal.etsi.org/Services/editHelp%21/Howtostart/ETSIDraftingRules.aspx)*[EDRs)](https://portal.etsi.org/Services/editHelp%21/Howtostart/ETSIDraftingRules.aspx),*

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](https://portal.etsi.org/Services/editHelp%21/Howtostart/ETSIDraftingRules.aspx) (Verbal forms for the expression of provisions).

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

# Executive summary

This present document specifies a high-level functional abstraction of PDL Smart Contract System Architecture. In particular, basic building blocks for designing, coding and testing Smart Contracts for the PDLs. This includes describing how different classes of systems interact with Smart Contracts. Processes, models, and detailed information are beyond the scope of the present document.

# Introduction

The present document defines a high-level functional abstraction of policies to design and code Smart Contract components. Smart Contracts are mere codes, and if not well planned, designed, coded and tested; can leave the system vulnerable to external attacks and internal errors.

# 1 Scope

**Scope of work to be undertaken:** The present document specifies the functional components of Smart Contracts, their planning, coding and testing. This includes:

a) reference architecture of the technology enabling Smart Contracts – the planning, designing and programming frameworks

b) specify how to engage using this architecture – the methods and frameworks the Smart Contracts building blocks possibly communicate

c) point out possible threats and limitations

# 2 References

## 2.1 Normative references

## 2.2 Informative references

# 3 Definition of terms, symbols and abbreviations

## 3.1 Terms

## 3.2 Symbols

## 3.3 Abbreviations

AML: Anti-Money Laundering

API: Application Programming Interface

SC: Smart Contract

CEN-CENELEC: European Committee for Standardization and European Committee for Electrotechnical Standardization.

DLT. Distributed Ledger Technology

EBP: European Blockchain Partnership

EBSI: European Blockchain Service Infrastructure.

EC: European Commission

EFTA: European Free Trade Association

eIDAS: Electronic Identification, Authentication and Trust Services.

EIRA: European Interoperability Reference Architecture

ESSIF: European Self Sovereign Identity Framework

ETSI: European Telecommunication Standards Institute

EU. European Union

FIG: International Federation of Surveyors

GDPR: General Data Protection Regulation

ICO: Initial Coin Offering

ICT: Information and Communications Technology

ISO: International Standards Organization

ITU: International Telecommunication Unit

KYC: Know Your Customer

OECD: Organization for Economic Co-operation and Development

PDL: Permissioned Distributed Ledger.

RA: Regulatory Authority

SG: Study Group.

SLA: Service Level Agreement

SME: Small and Medium Enterprise.

STO: Security Token Offering.

TOOP: The Once-only Principle

TSAG: Telecommunication Standardization Advisory Group.

UN/CEFACT: United Nations Centre for Trade Facilitation and Electronic Business.

UNCITRAL: United Nations Commission on International Trade Law.

UNE: Spanish Association for Standardization.

# Smart Contracts

##  Definition

Any PDLT general goal is the distributed managing of a common data repository defining a current global state. There is no assumption on the type of data stored. When such data is executable code, then the induced global state can be seen as the state of a distributed virtual machine. Any executable code stored on a PDLT is dubbed a “Smart Contract” (SC).

As per ISO definition, Smart Contract is:

***It is a computer program, stored in a distributed ledger system, wherein the outcome of any execution of the program is recorded in the distributed ledger.***

## Smart Contract Programming Paradigms

In general, any executable code deployed on a PDLT can be considered a SC, but different schemes are possible to achieve this. In the next two sections we will show the two main proposals adopted in practice, highlighting their similarities to classical programming languages paradigms.

### Object-Oriented Paradigm

Historically the main model adopted for SCs has been on the line of the traditional Object Oriented paradigm. As such an SC is seen as a code entity composed of two main sections:

* Internal storage, in the form of identifiers – value associations akin to a dictionary, similarly to object fields;
* Functions’ definitions, specifying the set of actions allowed for the given SC with the appropriate scope modifiers, similarly to object methods.

A general contract definition, i.e. the writing of a SC code, is akin to defining a new class. A new SC can then be deployed to the PDLT, akin to an object being instantiated from a class. Similarly to how an object has a reference, so a SC, once deployed, holds a unique identifier. The deploy operation of a new SC updates the global state, making all its fields and functions visible and callable, according to their respective visibility modifiers. Any accessible function of deployed SCs can then be called as many times as wanted, according to their operational logic specified by their code.

This model of SCs generally describes all PDLT protocols derived from or inspired by the Ethereum protocol, such as the Quorum protocol form JPMorgan[[1]](#footnote-1) and Move for LIBRA[[2]](#footnote-2).

### Properties

Since SCs code resides on-chain, any SC inherits all of the underlying PDLT properties. However, being executable code, such properties get new meanings.

#### Immutability

As any data on a PDLT, a SC is immutable. This means that the code deployed on the PDLT, specifying the contract logic, cannot be changed. Do note that the values contained inside a SC internal storage are mutable as expected. A consequence of Immutability is *Immortality*, since, once deployed, a SC cannot be removed. However, most PDLT protocols allow for a logic marking of a SC as *disabled*, to disable future calls to a given SC. In order to disable a SC needs to specify an appropriate function inside its code.

#### Availability

An SC is always available as long as the underlying PDLT is accessible. This means that a SC functions can be invoked, and fields can be read, as long as an entity has the appropriate privileges specified by the contract and PDLT.

#### Transparency

Any entity, with the appropriate privileges, might inspect a SC code and current values. As such, it is transparent to all intended participants what the logic that the SC enforces is. Moreover, any call to a function of a contract is performed through a general state update on the PDLT (i.e. transaction). As such, all function calls are recorded and traceable, providing *Auditability*.

#### Self-Execution

Any execution of a SC, i.e. an invocation to one of its visible functions, is performed by the PDLT, not by the user invoking the SC, nor by the SC creator. The SC execution is protected by the distributed consensus of the PDLT, as such, it is behind the control of any single party, and cannot be tampered with by them. This property induces the sub-properties of:

* *Atomicity*: a SC invocation runs entirely or fails without affecting the state;
* *Synchronicity*: a SC invocation is executed in a synchronous way;
* *Determinism*: a SC invocation must return the same result for any party executing it.

#### Reusability

SCs are coded once and executed multiple times. To ensure reusability, the contracts are advised to be generalised; this will enable the standardisation of contracts for industries. For example, the SC for cellular service is standardised with required field for QoS metrics such as latency; all the operators, in this case, will be required to specify the latency they will provide.

## Components

## Storage

Smart contracts are mostly stored in distributed ledgers; however, their storage depends upon the nature of ledger architecture. For example, in case of a public blockchain such as Ethereum, a smart contract will be stored by all nodes; on the contrary, in a permissioned blockchain such as Hyperledger, smart contracts are installed on the channels (an abstract point-to-point link between nodes) which are established through communication between nodes.

## Life cycle of a Smart Contract

Diagram

#  Smart Contracts – Planning, coding and Testing

Smart Contracts are software programs; the inherent properties such as auto-execution can be dangerous and small inaccuracy can produce erroneous executions. Such as, with a smart intentional or unintentional, logical or syntax error can make a SC can go in an infinite loop. Hence, SC should be planned, coded and tested thoroughly before deployment.

## Planning phase

### Verification

Smart Contracts should be tested for every step of execution. TBD

### Validation

The smart contract should be the exact and true representation of natural language contract and should perform only the tasks specified in natural language contract. The most pertinent discussion here is that: does the smart contract truly and completely encode the legal contract logic? Is the software-code water-tight enough to prevent the unwanted execution of the contract by unauthorised parties?

## Coding Phase

## Testing Phase

The Smart Contracts’ testing varies from traditional software testing in several ways; In traditional software which is mostly meant to be coded for a limited number of users, and the executed on a single machine at a time. On the contrary, SCs is executed once and the distributed ledger comprising of several machines get updated with the results. This makes the task of

## Modelling

TBD - traditional modelling methods such as ER and UML diagram can be modified to fit in with SCs?

#  Architectural requirements for Smart Contracts

## Reference Architecture

## Interaction between PDLs

It can be foreseen as that most of the major organisations will be adopting PDLs to maintain their company records. For example, a telecom company can be using a PDL to maintain its billing records, an automobile company for their vehicle specifications, and an aviation authority to log their flight records. The next step would be these ledgers should have access, to some extent, to other ledgers. For example, a tractor should be able to records its start and end of a trip to the logistic company ledger but should not be able to change any record. However, this tractor can have full access to its own company’s PDL where it can record and access its maintenance information such as service history.

Hence, multiple PDL should be able to inter-connected by a secure Access Control (AC) mechanism and should allow permissioned access to only certain sections with appropriate AC credentials. PDLs should be able to access each other’s ‘related and concerned only’ records only such as GP Surgery should be able to access person’s health records but don’t have access to their financial transactions.

If a PDL wishes to access an object from other PDL, in this case, we consider examples of ORG1 – PDL is the PDL wishes to access the information(the ‘Subject)’, the data it wants to access from other PDL(i.e. ORG2 - PDL) is called as ‘object’. The components are explained in detail below:



Figure : Overall two PDLs with respective MSP

### Membership Service Provider (MSP)

The term MSP is adopted from Hyperledger Fabric and its function here is similar (to some extent) to HLF’s MSP. Every PDL has one ‘Membership Service Provider’(MSP) – and is responsible to maintain node IDs, their access rights and roles. All MSPs keep their own ledgers for identity management, to which access can be granted to other PDL-MSPs for verification. The common records (i.e. access of MSP2 from MSP1) should be recorded, and accessible by both.

### Smart Contracts (SC)

MSP will install three different Smart Contracts (SC) in MSP ledger; one to ensure the dynamic up of public keys, one to amend access rights of the nodes (which is accessible only by the MSP itself or nodes with higher authority) and other to grant access to other PDLs.

When a node/peer wishes to update its public key, it will locally generate Public Key/Private Key pair and invoke the smart contract to update this credentials at MSP end. A peer on its own cannot amend, its role and access rights; this is still managed and controlled by the MSP or the authorised node.

### Node Identity

The node identities and access rights are kept by MSP, in a separate ledger; when the access rights or ID of the node is changed, a smart contract is executed by the authorised node and the state changed is recorded in MSP ledger.

### Inter-PDL communication

Two PDLs should be able to connect through MSPs only and as MSP has no access to ledger data, this access should ask by a designated node (Admin node (AN)). When a PDL (i.e. ORG1-PDL) wishes to access data from other PDL (i.e. ORG2-PDL), the AN must ask permission from its local MSP (i.e.MSP1). The MSP then contacts the other MSP (i.e. MSP 2 in this example) by sending Access Request (AR). AR must define metrics of access (i.e. PDL ID, roles, duration and PKs of the nodes).



Figure : MSP1 sends AR to MSP2

MSP2 grants access to its PDL by invoking SC; recording the request and access grant to its ledger and sending the Transaction ID to MSP 1.



Figure : MSP confirms the access with Transaction ID

MSP2 will subsequently record the keys of MSP1 and assign the role (such as visiting peers with read-only access). As soon as the duration elapsed, the access is revoked, and all the ledgers are updated.

#  Smart Contracts – Applications, solutions and Needs

## Regulatory Aspects

## Security of the Contracts

## Enforceability

Every contract or agreement should be enforced and respected; the contracts which cannot be enforced has no significance. Enforceability in SCs can be achieved by two methods: by embedding the conditions in the code, and by monitoring by the Regulatory Authority (RA). These two conditions are inter-related and should be achieved by the supervision of the regulatory authority.

### The role of the regulatory authority:

A Regulatory Authority can monitor one or many PDLs depends upon the industry requirements. Such as a group of car manufacturers, each managing their own PDL may wish to form a consortium monitored and managed by a RA. The presence of a RA makes the participants of the consortium to follow code-of-conduct and the SCs to be respected. In the situations, where either of the party doesn’t comply with the agreement RA can step in and resolve the dispute. However, to ensure the data privacy, RA should have only controlled access the ledgers’ data and should get involved only in the occurrences of disputes between the participants.

Diagram

#  Limitations of Smart Contracts

##  Inter and Intra system threats

##  Limitations

Annex B:
Title of annex

# B.1 First clause of the annex

## B.1.1 First subdivided clause of the annex

<Text>.

<PAGE BREAK>

Annex <L>:
Authors & contributors

The following people have contributed to the present document:

**Rapporteur**:
Title, Firstname, Lastname, company

**Other contributors**:Title, Firstname, Lastname, company

<PAGE BREAK>

Annex <L+1>:
Bibliography *(style H9)*

The "Bibliography" annex identifies additional reading material not mentioned anywhere in an ETSI deliverable including annexes. These publications might or might not be publicly available (no check is made by the Secretariat).

The Bibliography **shall include** list of standards, books, articles, or other sources on a particular subject which are not cited anywhere in an ETSI deliverable including annexes.

The Bibliography **shall not include** *documents listed in clauses 2.1 and 2.2.*

*[ETSI Drafting Rules](https://portal.etsi.org/Services/editHelp%21/Howtostart/ETSIDraftingRules.aspx)* [(](https://portal.etsi.org/Services/editHelp%21/Howtostart/ETSIDraftingRules.aspx)*[EDRs)](https://portal.etsi.org/Services/editHelp%21/Howtostart/ETSIDraftingRules.aspx),* clause 2.14.

* Use **Heading 9** style for the "Bibliography" annex, see clause 2.13 of the [EDRs](https://portal.etsi.org/Services/editHelp%21/Howtostart/ETSIDraftingRules.aspx) for examples.
* For the listed material use the **Normal** style or bulleted lists (e.g. **B1+**), do not use numbered references.

*EXAMPLE 1:*

*<*Publication*>*:"*<*Title*>".<*Edition*>*. *<*Year*>*, *<*Issue designation*>*, *<*Page location*>*. *(style Normal)*

WEAVER, William. "Command performances". December 1985, vol. 42, n° 12, p. 126-133). *(style Normal)*

*EXAMPLE 2:*

* <Publication>: "<Title>". *(style B1+)*
* ETSI EN 300 066: "ElectroMagnetic Compatibility and Radio Spectrum Matters (ERM); Float-free maritime satellite Emergency Position Indicating Radio Beacons (EPIRBs) operating in the 406,0 MHz to 406,1 MHz frequency band; Technical characteristics and methods of measurement". *(style B1+)*

<PAGE BREAK>

1. https://docs.goquorum.com/en/latest/ [↑](#footnote-ref-1)
2. https://developers.libra.org/docs/assets/papers/libra-move-a-language-with-programmable-resources.pdf [↑](#footnote-ref-2)