ETSI GR PDL - v0.0.4 (2020-02)

**Group Report**

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

Release

***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.

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 5

Introduction 5

1 Scope 5

2 References 5

2.1 Normative references 5

2.2 Informative reference 5

3 Definition of terms, symbols and abbreviations 5

3.1 Definitions 5

3.1.1 Smart contract 5

3.1.2 On-chain Smart Contract 5

3.1.3 Off-chain Smart Contract 5

4 Abbreviations 6

5 Introduction to Smart Contracts 7

5.1 Smart Contract Programming Paradigms 7

5.2 Introduction 7

5.2.1 Object-Oriented Paradigm 7

5.2.2 Properties 7

5.2.2.1 Immutability 7

5.2.2.2 Availability 7

5.2.2.3 Transparency 7

5.2.2.4 Self-Execution 8

5.2.2.5 Reusability 8

5.3 Storage 8

5.4 Life cycle of a Smart Contract 8

6 Smart Contracts – Planning, coding and Testing 9

6.1 Introduction 9

6.1.1 Planning phase 9

6.1.1.1 Draft template 9

6.1.1.2 Terms Negotiation 9

6.1.1.3 Compile Draft 9

6.1.1.4 Review 9

6.1.2 Coding and Testing phase 9

6.1.2.1 Code Verification 9

6.1.2.1.1 SC Analysers 9

6.1.2.1.2 Testbeds 9

6.1.3 Validation 9

6.1.3.1 Code Testing 9

6.1.3.2 Deployment and Execution Phase 10

7 Architectural requirements for Smart Contracts 10

7.1 Reference Architecture 10

8 Smart Contracts – Applications, solutions and Needs 10

8.1 Regulatory Aspects 10

8.2 Security of the Contracts 10

8.3 Enforceability 10

8.3.1.1 Introduction 10

8.3.1.2 Expected role of the regulatory authority 10

9 Limitations of Smart Contracts 10

9.1 Inter and Intra system threats 10

9.2 Limitation 10

Annex A: Title of annex 12

Annex B: Title of annex 13

B.1 First clause of the annex 13

B.1.1 First subdivided clause of the annex 13

Annex C: Bibliography 14

Annex : Change History 15

History 16

# 

# 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!/Howtostart/ETSIDraftingRules.aspx)* [(](https://portal.etsi.org/Services/editHelp!/Howtostart/ETSIDraftingRules.aspx)*[EDRs)](https://portal.etsi.org/Services/editHelp!/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!/Howtostart/ETSIDraftingRules.aspx)* [(](https://portal.etsi.org/Services/editHelp!/Howtostart/ETSIDraftingRules.aspx)*[EDRs)](https://portal.etsi.org/Services/editHelp!/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!/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.

# 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

# References

## Normative references

## Informative reference

# Definition of terms, symbols and abbreviations

## Definitions

### Smart contract

As per ISO /TR23455/2019 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 on the distributed ledger.***

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

In this document we separate smart contract and smart legal contract.

### On-chain Smart contract

### Off-chain Smart contract

# 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.

# Introduction to Smart Contracts

## Introduction

In general, an computer program deployed on a PDL 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.

## Smart Contract Programming Paradigms

Any PDL 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 PDL is dubbed a “Smart Contract” (SC).

### Object-Oriented Paradigm

Historically, the main model adopted for SCs has been on the line of the traditional Object Oriented paradigm. As such, a 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 PDL, 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 PDL protocols derived from or inspired by the distributed consensus protocol Ethereum protocol, such as the Quorum protocol from JPMorgan[[1]](#footnote-1) , Tezos or stellar protocol etc.

### Properties

Since SCs code resides both on-chain or off-chain, and inherits most of the underlying PDL properties. However, being executable code, such properties get new meanings:

#### Immutability

As any data on a PDL, a SC is immutable. This means that the code deployed on the PDL, specifying the contract logic, cannot be changed. 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 PDL 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 PDL 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 PDL.

#### 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 PDL (i.e. transaction). As such, all function calls are recorded and traceable, providing *Auditability*. As a further remark, we observe how the code of the contract is itself auditable given a source code and a certain compiler, i.e. any user can re compile the source code to make sure that the object code actually executed on ledger corresponds.

#### Self-Execution

Any execution of a SC, i.e. an invocation to one of its visible functions, is performed by the PDL, not by the user invoking the SC, nor by the SC creator. The SC execution is protected by the distributed consensus of the PDL, 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 returns the same result for any node executing it.

#### Reusability

SCs are coded once and executed multiple times. A given smart contract can be uses as template for a wider set of applications sharing the same high level logic. The actual behaviour of a given contract may change depending on the parameters which are set at invocation time. For example, the SC for cellular service is modelled with required fields for QoS metrics such as latency; all the operators, in this case, will be required to specify the latency they will provide.

## Storage

Smart contracts are 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 stored only on the nodes that are part of a given channel (an abstract point-to-point link between nodes) and are established through communication between nodes.

## Life cycle of a Smart Contract

A smart contract is a computer program, the difference is that the smart contracts are immutable, so it requires great care to program them and should be tested on several levels before deployment. This section presents the recommended life cycle a smart contract should follow in order to avoid the dangers of mis-programming.

A screenshot of a social media post

Description automatically generated

# Smart Contracts – Planning, coding and Testing

## Introduction

Smart Contracts are software codes, and similar efforts are required to program them like any other software program. The difference is, however for any usual software, the bugs can be fixed in new releases or through software patches, but SCs stored in a DLT by design lack this provision. The deployed SC code becomes ossified to the system and further changes to the same code under the same identification is not permitted. Hence, the careful planning and scrutiny of the code before deployment to the ledger is of utmost importance.

### Planning phase

#### Draft template

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

#### Terms Negotiation

#### Compile Draft

#### Review

### Coding and Testing phase

#### Code Verification

Smart Contracts should be tested at every step of execution. There are methods to inspect the SC code are discussed as follows:

##### SC Analysers

There are number of open-source SC analyzers such as Securify and SmartCheck are available to analyse the SC code and tag the vulnerabilities present in the program. These vulnerabilities such as unintiated functions, can provide third-party(possibly malicious) access to the contract, thus to the ledger. These analysers prevent external accesses by inspecting the code and flagging the possible vulnerablabilities in the code. However, all the analysers have their limitations, also the attacks on the contracts are evolving, hence more comprehensive scrutiny of the code can be achieved by multiple analyses techniques. Another important consideration for analyser is the support for ledger technology, most of the available analysers are for Ethereum and Hyperledger and the adopters of the ledger should look for their own supported analyser.

##### Testbeds

Ethereum Ropsten, Rinkeby etc. Private nodes, Hyperledger Fabric dockers, University of surrey test bed.

Sandbox approaches

#### Validation

The smart contract should be the exact and true representation of natural language contract and should perform only the tasks specified in the natural language contract. In other words, semantic gap[Making SC smarter] that is the expected execution and the actual execution must be avoided in order to avoid the wrong doings of the contract and implement an error-free code.

The most pertinent discussion here is that: does the smart contract truly and completely encode the contract logic? Is the software-code water-tight enough to prevent the unwanted execution of the contract by unauthorised parties?

#### Code Testing

Refer to traditional software testing – mirror in the context of SC

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

### Deployment and Execution Phase

#### Deployment

#### Execution

#### Termination

# Architectural requirements for Smart Contracts

## Reference Architecture

# Smart Contracts – Applications, solutions and Needs

## Regulatory Aspects

## Security of the Contracts

## Enforceability

### Introduction

Comments: contracts are not supervised by any regulator – enforceability is achieved by mutual trust between the operators.

Smart contracts – can become sc legal contracts depends on the situations – thus enforceable - mechanism etc.

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

Several regulators can be involved in this eco-system –

### Expected role of the regulatory authority

A Regulatory Authority is expected to monitor one or many PDLs depending on 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 does not comply with the agreement RA is expected to help resolving the dispute. However, to ensure the data privacy, RA should only have controlled access over the ledgers’ data and can get involved only in the occurrences of disputes between the participants.

Diagram

# Limitations of Smart Contracts

## Inter and Intra system threats

## Limitation

# 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 C:  
Bibliography

Annex :  
Change History

| Date | Version | Information about changes |
| --- | --- | --- |
| 11-2019 | 0.0.1 | Initial draft – added table of contents |
| 12-2019 | 0.0.2 | Added SC introfuction, types and some text |
| 01-2019 | 0.0.3 | Cleaned draft |
| 02 - 2020 | 0.0.4 | Cleaned up after F2F meeting – deleted section 6 |
|  |  |  |
|  |  |  |

# Document History

|  |  |  |
| --- | --- | --- |
| **Document history** | | |
| <Version> | <Date> | <Milestone> |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

1. https://docs.goquorum.com/en/latest/ [↑](#footnote-ref-1)