ETSI GR PDL 004 V1.1.1 (2021-02)

Permissioned Distributed Ledgers (PDL)
Smart Contracts
System Architecture and Functional Specification

Disclaimer

Reference

DGR/PDL-004_smart contract

Keywords

blockchain, policies, PDL, SLA, smart contract

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

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

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Contents

Intellectual Property Rights

Essential patents

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

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

Trademarks

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

Foreword

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

Modal verbs terminology

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

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

Executive summary

The 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

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

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] ACM Digital Library: "Securify: Practical Security Analysis of Smart Contracts".

NOTE: Available at https://dl.acm.org/doi/pdf/10.1145/3243734.3243780.

[i.2] ACM Digital Library: "SmartCheck: Static Analysis of Ethereum Smart Contracts".

NOTE: Available at https://dl.acm.org/doi/pdf/10.1145/3194113.3194115.

[i.3] ITU-T Report: "Distributed Ledger Technologies and Financial inclusion".

NOTE: Available at https://www.itu.int/en/ITU-T/focusgroups/dfs/Documents/201703/ITU_FGDFS_Report-on-DLT-and-Financial-Inclusion.pdf.

[i.4] ETSI GR PDL 003: "Permissioned Distributed Ledger (PDL); Application Scenarios".

NOTE: Available at https://portal.etsi.org/webapp/WorkProgram/Report_WorkItem.asp?WKI_ID=57511

[i.5] United Nations Commission on International Trade Law.

NOTE: Available at https://uncitral.un.org/

[i.6] Decentralized Public Key Infrastructure.

NOTE: Available at https://medium.com/hackergirl/decentralized-public-key-infrastructure-4e7ea9173bac

3 Definition of terms, symbols and abbreviations

3.1 Terms

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

coin: implementation using a unique ledger and usually used for financial transactions (e.g. Ether, Bitcoin)

eternal contracts: contracts which are active for infinite time

mainnet: ledger in-production

NOTE: The contracts and transactions on a mainnet are ultimate.

master-chain: primary chain where the executions of the Smart Contract are recorded

off-chain smart contract: smart contracts stored away from the ledger (i.e. trusted database or side-chain) and their execution may depend on on-chain contracts (i.e. on-chain contract can initiate off-chain contracts) and later the state can be updated

on-chain smart contract: contract that resides in the master-chain and on side-chain, that is executed directly without the instantiation of any other contract

NOTE: The beneficiaries get rewarded as soon as the contract is executed without the involvement of any other contract.

participants: participants are the members of the PDL which keep the copy of the ledger and take part in the consensus

Ricardian contract: single contract document which is both easily readable by human and machines and not self-executable

NOTE 1: It is formatted as a text file and digitally signed by the issuer of the contract.

NOTE 2: The security of a Ricardian contract is achieved by OpenPGP and all the signing keys are included within the contract so eliminates the use of external certificate authority, in other words a Ricardian contract carries its own PKI with them.

NOTE 3: The Difference between Ricardian contract and Smart Contract: The major difference between the Smart Contract and the Ricardian contracts is that Smart Contracts are executable code but Ricardian contracts are the agreements recorded in a single file and not executable on their own. A Smart Contract does not have to be a Ricardian contract and a Ricardian contract is not a Smart Contract, but a Smart Contract can execute a Ricardian contract.

side-chain: chain(s) which work as a secondary chain to the main-chain/ledger

NOTE: It can be used to off-load some of the computations for scalability or privacy.

Smart Contract (SC): computer program stored in a distributed ledger system, wherein the outcome of any execution of the program is recorded on the distributed ledger

NOTE: A Smart Contract might represent terms in a contract in law and create a legally enforceable obligation under the legislation of an applicable jurisdiction. A Smart Contract may but does not have to be human readable and is self-executable. Any executable code stored on a PDL is dubbed a "Smart Contract" (SC).

The focus of the present document is Smart Contract as software codes and is different from legal contracts.

stakeholders: parties that benefit from the PDL

NOTE: All the stakeholders may or may not keep the copy of the ledger (i.e. act as a node) and take part in consensus.

testnet: ledger or sandbox on which Smart Contracts can be installed to test their working and performance prior to installation on a mainnet

NOTE: Testnets are installed to test the performance of the code and the transactions and Smart Contracts are for the testing purposes only.

Table 3-1: Comparison of Ricardian and Smart Contract

Contract Type	Machine-Readable	Human-Readable	Self-Executable
Ricardian Contract	Yes	Yes	No
Smart Contract	Yes	Optional	Yes

3.2 Symbols

Void.

3.3 Abbreviations

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

API	Application Programming Interface
DLT	Distributed Ledger Technology
ETSI	European Telecommunications Standards Institute
GDPR	General Data Protection Regulation
ICT	Information and Communications Technology
ITU	International Telecommunication Unit
PDL	Permissioned Distributed Ledger
QoS	Quality of Service
SC	Smart Contract
SLA	Service Level Agreement
TEE	Trusted Execution Environment
UNCITRAL	United Nations Commission on International Trade Law

4 Introduction to Smart Contracts

4.1 Introduction

A Smart Contract is a computer program deployed on a PDL. The primary purpose of smart contract to keep certain software in a PDL that execute on certain execution requests.

Any PDL's general goal is the distributed management of a common data repository defining a current global state; there is no assumption on the type of data stored. When such data is an executable code (i.e. smart contracts), the induced global state can be seen as the state of a distributed virtual machine.

4.2 Object-Oriented Paradigm

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

- Internal storage, in the form of identifiers - value associations akin to a dictionary, similarly to object fields.
- Functions' definitions, specify the set of actions allowed for the given SC with the appropriate scope modifiers, similarly to object methods.

Similar to the concepts of Object-Oriented programming a Smart Contract is instantiated from a class, and once instantiated holds a unique identifier; that is to say every instantiation is unique. The deployed Smart Contract holds a global state which means that all its fields and functions become visible and callable by other contracts (depending on access rights). Moreover, a deployed Smart Contract can be called as many times as required; however, this is dependent on the implementation.

Smart contracts have different implementations depending on the technology and consensus mechanism such as PDL types (e.g. Hyperledger, Quorum)

4.3 Properties of Smart Contracts

4.3.1 Introduction

The properties of Smart Contracts directly depend on the properties of the underlying PDL and some properties due to their requirements.

4.3.2 Immutability

As any data on a PDL, an SC is immutable; this means that a Smart Contract code, once accepted through consensus, cannot be changed. However, modifications through other methods such as proxy contracts or introducing a new Smart Contract, are possible. In such an event, the old version of the contract remains in the chain. A consequence of immutability is Importability which means that it cannot be deleted from the ledger after deployment. This brings the challenges of scalability as a PDL might be populated with dormant contracts over time. The details on scalability are discussed in later clauses.

The values contained inside an SC's internal storage are mutable as expected through function calls; for example, in an auction contract bid values will change with new bids but the final winning bid may be immutable.

4.3.3 Availability

In the case of on-chain SC, it is always available as long as the underlying master ledger is accessible. This means that a SC function can be invoked, and its fields (i.e. variables) can be read, by an entity as long as the entity has the appropriate privileges specified by the contract and the PDL. However, in the case of off-chain Smart Contracts, if the ledger where the contract is installed (i.e. secondary PDL) is not available, the SC is not accessible by the master PDL.

4.3.4 Transparency

Any entity, with the appropriate privileges, might inspect a SC code and current values. As such, it is transparent to all intended participants of the PDL. Transparency is not to be confused with immutability; a contract code remains unchangeable even though it is transparent to both parties.

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 in the PDL and traceable by the members of the PDL with appropriate access rights.

4.3.5 Self-Execution

Any execution of a SC, i.e. an invocation to one of its visible functions, is performed by the PDL nodes, 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 beyond the control of any single party to execute a Smart Contract without the approval of PDL members. This property induces the sub-properties of:

- Atomicity: an SC invocation runs entirely or fails without affecting the state (i.e. there is no such thing as partial SC execution).
- Synchronicity: an SC invocation is executed in a synchronous way (i.e. every member with appropriate access rights get the update).
- Determinism: an SC invocation returns the same result for any node executing it.

4.3.6 Reusability

SCs are coded once and can be executed multiple times depending on PDL governance. A given Smart Contract can be used as a 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 telecom operators, in this case, will be required to specify the latency as a parameter.

4.4 Storage

Smart Contracts are typically stored in distributed ledgers; however, their storage depends upon the nature of the ledger architecture. For example, in case of a permissionless 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.

For off-chain SCs, the contracts may be stored on a trusted data storage, away from the ledger. This type of storage mechanism needs special security measures set out by the governance of the PDL.

Reusability techniques such as template contracts can be used to allow efficient storage of the contracts. The decision of storage is dependent on the implementation of a PDL, and the technology the companies adopt. The limitations due to the external existence of a contract is discussed in clause 8.

4.5 The Lifecycle 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 is good be tested on several levels before deployment. This clause presents the recommended lifecycle, a Smart Contract may follow in order to avoid the dangers such as erroneous code. This recommended lifecycle consists of three phases: planning phase, coding & testing phase and deployment & execution phase. The phases are explained in detail in clause 5.

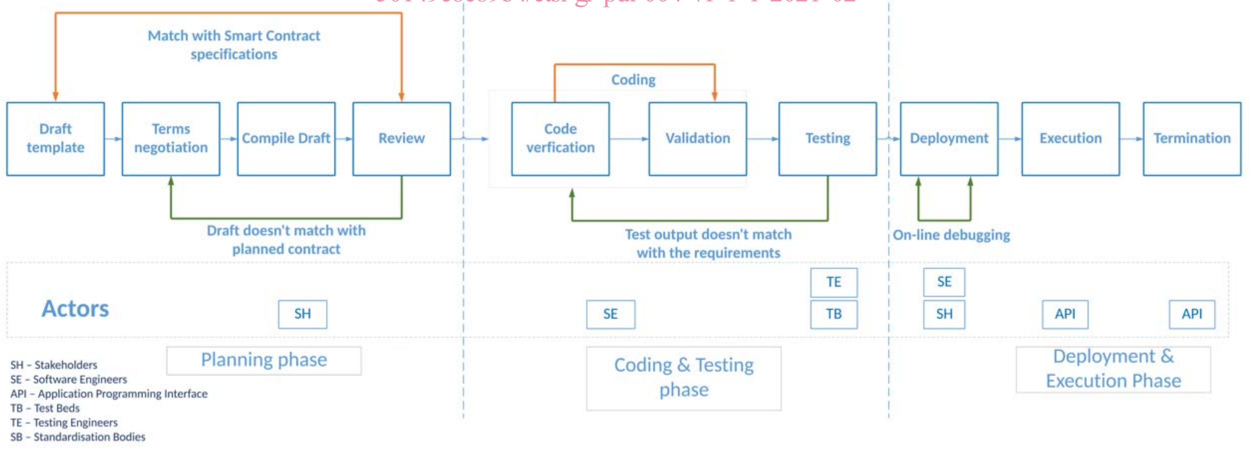

Figure 4-1: Lifecycle of a Smart Contract

5 Smart Contracts - Lifecycle phases

5.1 Introduction

Smart Contracts are software codes similar to any other software program. The difference from usual software is in the way the bugs are being fixed. The nature of PDL, does notallow backward modification of information or code so any change to Smart Contract can only be applied to the time of deployment and onwards.

Hence, careful planning and scrutiny of the code before deployment to the ledger is of utmost importance. In this clause, the stages of the Smart Contract lifecycle (Figure 4-1) are defined, which the industries may follow to implement Smart Contracts in the adopted PDL.

5.2 Planning Phase

5.2.1 Introduction

A Smart Contract can be deployed in many ways, and the deployment methods are dependent on the underlying ledger technology and acceptable by the participants through consensus. The goal is to create a contract that can be trusted by participants who do not trust each other. The planning of a Smart Contract will enable the participants to define their requirements and functionalities of a Smart Contract. The planning phase may include:

1) governance - ownership and access rights;
2) design - coding and testing;
3) deployment; and
4) management planning.

5.2.2 Governance

5.2.2.1 Introduction

A Smart Contract may define a contract, and its associated terms and conditions covering the full lifecycle of the contract, between the participants. Governance planning defines the authority of different stakeholders over the contract, for example, ownership and access rights.

Usually, the creator of a contract is the owner as well; the owner of the contracts has exclusive privileges such as contract destruction. However, in PDLs where contracts can be reused by several participants for several unrelated transactions, it is feasible to have a role-based ownership mechanism. In Role-Based ownership, the operations of a contract are governed by a group of participants with appropriate privileges; as PDL is a collaborative ledger, these privileges can be specific to a contract.

5.2.2.2 Single-party Governance

The Smart Contract, when deployed, is usually identified as being governed by a specific part (N=1) or a group of distinct parties depending on consensus and governance model.

This agreement needs to take into account the legal and business aspects of the Smart Contract, and address issues such as who is eligible to stop, terminate, or upgrade the Smart Contract, and how these are enforced contractually or technically.

Smart Contracts are a digital model of such contracts, and actors and their arrangements are beyond the scope of the present document.