This is an agreed use case and set of requirements from the Hyperledger Requirements WG. This document was first drafted on 7 June 2017, it has gone through several rounds of review at the Hyperledger RequirementsWG, it has been shared with the Hyperledger IdentityWG and the Hyperledger ArchitectureWG for discussion, and it was formally approved on 24 July 2017.

Hyperledger - After-the-fact mandate changes

Author: Dr. M. Oskar van Deventer, Rieks Joosten

Reviewers: Clive Boulton, Tracy Kuhrt, Aaron Benningfield,
Andreas Freund, Mark Parzygnat, Steve Anon

Oskar.vanDeventer {Senior Scientist Blockchain, TNO} @ TNO.nl
Rieks.Joosten {Senior Scientist, TNO} @ TNO.nl

Section 1 - Intro

Overview of the Business Problem or Opportunity

(Please explain the business problem or opportunity in business terms without technical jargon, and without mention of a distributed ledger.)

A smart contract (a.k.a. chain code, blockchain applications) typically specifies the identities of the persons, organisations or entities that are entitled to act upon the smart contract, e.g. for oracle-type input to the smart contract and/or execution of the smart contract. There are several use cases (see below), in which the mandate for the smart contract of one's access control list (ACL) needs to be changed after-the-fact, even if the smart contract itself does not need to be executed. Many of these cases are legally enforceable. It is a business problem, if a platform that executes smart contracts cannot comply with a legally-enforced after-the-fact mandate change.

Current Solution

(If there are systems in place today which automate the above business problem/opportunity, please explain what exists.)

Current monolithic (non-DLT) systems do not have this problem, as a single party is responsible for the system. Hence this single party is able to comply with the legally-enforceable request.

Why Distributed Ledger Technology (DLT)?

(Please explain how distributed ledger technology would improve the current solutions (if they exist) or enable new solutions which were previously unavailable. The goal here is to ensure we do not have a solution looking for problems, but problems where solutions would become possible or significantly improved by using distributed ledger technology.)

The reason that consortia use DLT ("consortium blockchain") is that the technology enables new forms of collaboration that does not require the creation of a central technical entity or trusted third parties.

Opportunity/Justification

(Use this section to give details that support the value of pursuing these user stories using distributed ledger technology. Some examples of information that might be included here are applicable market segments, workloads, user bases, etc. and any associated data.)

Running DLT together does not make a consortium immune to legally-enforceable requests for after-the-fact mandate changes. If a consortium is technically unable to comply with the request, this could result in penalties, confiscation of servers and ultimately even in the dismantling of the consortium.

Section 2 - User Stories and Requirements

(Please explain this use case using actors and interactions and assuming the ideal distributed ledger technology exists.)

2.1 Mandate with user consent

Alice is having problematic debts. The court has assigned Bob as legal guardian, who has the authorization to perform financial transactions on behalf of Alice. The court has also declined Alice the authorization to perform financial transactions herself. This includes both her bank account and a bunch of smart contracts running on DLT. Being a law-abiding citizen, Alice complies. She instructs her bank about the new situation, and she signs over the authorization over the smart contracts to Bob.

There are several variations to this use case, all leading to the same requirement. In another variation, Alice is a convicted criminal, and the court orders the confiscation of her illegally-obtained assets, including smart-contract-controlled ones. In all cases, the smart contract did not anticipate this case, the smart contract is not required to get executed, Alice is required to hand over the authorization over the smart contract, and Alice decides to comply.

In some cases, Alice would keep the authorization over the smart contract, but she want to include Bob. For example, Alice accepts Bob as new business partner, Alice marries Bob, or Alice accepts Bob as her caretaker.

Requirement 1: A user ("anchor") shall be able to sign over its authorization over a smart contract to another user ("guardian") after the fact. In this requirements we see the following two scenarios that could play out.

1a: The original user ("anchor") loses its authorization over the smart contract.

1b: The original user ("anchor") maintains its authorization over the smart contract.

2.2 Mandate with consortium consent

Cullen has died. The heirs have assigned Diana as their testamentary executor. Diana discovers that Cullen has left some of his assets under the control of a smart contract. Diana contacts the blockchain consortium (consortium running a DLT together) that executes Cullen's smart contract, and requests the control over Cullen's smart contract. Being a law-abiding consortium, they comply. The consortium signs Cullen's smart contract over to Diana's credentials.

There are several variations to this use case, all leading to the same requirement. In another variation, Cullen has dementia and the court has assigned Diana as his legal guardian. Or Cullen is a convicted criminal, unwilling or unable to comply to a court order to hand over control over a smart-contract-controlled assets. In all cases, the smart contract did not anticipate this case, the smart contract is not required to get executed, Cullen is unwilling or unable to hand over the authorization over the smart contract, the blockchain consortium is requested to perform the authorization hand-over, and the blockchain consortium decides to comply.

Another variation is where Cullen has lost control over the identity that allows him to use the smart contract, e.g. the private keys by which he wields the control are lost or stolen. In this case, the authorization over the smart contract must be reassigned to a new identity of Cullen.

Requirement 2: A blockchain consortium shall be able to sign over the authorization over a smart contract from one user ("anchor") to another user ("guardian").

Here is a more detailed elaboration of the second requirement

A blockchain consortium should create and maintain a set of conditional authorizations (e.g. on the blockchain) that allow specific parties to reassign the authorization over smart contracts from one user to another, where each such authorization is associated with one (possibly complex) condition that specifies the scope of the authorization.

Conditions are meant to impose constraints on the applicability of the authorization. For example, it may restrict the (kinds of) smart contracts and/or the (kinds of) parties to which it applies.

Conditional authorizations thus cater for different governance models, jurisdictions, other legal or regulatory constraints, business policies etc. Also, court orders that require the signing over of contracts can then be catered for. The realization of conditional authorization (contracts) is considered outside the scope of this document.

Each of these user stories may have a legal addendum, in which some of the consortium partners, or the blockchain consortium as a whole is taken to court. There may be appeals, etcetera, but the final outcome is that the court orders the partners or consortium to comply, enforced by the threat of penalty payment, loss of members, regulatory pressures and ultimately dismantling of the consortium.

Jurisdiction may be a complicating factor if the different partners of the blockchain consortium reside in different countries, which is left out of scope of the present document.

2.3 Mandate for one, multiple or all smart contracts

In some cases, the signing over of authorization is for a single smart contract. In other cases, it is for multiple smart contracts at the same time, or even for all smart contracts of the original user. It may be inefficient if all smart contracts need to be signed over one by one, and it might be more efficient if it would be possible to sign over the authorization of multiple or all smart contracts for a single user at once.

Requirement 3: It should be possible to re-assign the authorization over multiple or all a smart contracts for a user ("anchor") to another user ("guardian") in a single action.

The requirement is formulated as a "should". It is not a need-to-have, and it could be resolved by an automated client application instead of a new blockchain feature.

2.4 Reversal of the mandate

Of course, none of the signing over needs to be permanent. New signing over back to the original "anchor" party could be performed, either voluntarily or enforced by court via the blockchain consortium.

Requirement 4: It should be possible to re-assign the authorization over a smart contract back to the original ("anchor") user.

The requirement is formulated as a "should". It is not a need-to-have, and it could be resolved by a new re-assignment of authorization.

Section 3 - Requirements Not Related to User Stories

(It is useful to specify requirements that should be considered but may not be apparent through the user story and usage examples.)

3.1 No handing over of private keys

These user stories interrelate with identity and the concept of self-sovereign identity. One apparent solution would be that Alice hands over the private keys to her identity, or that the blockchain consortium creates a new set of private keys for Cullen. Such solution would be unacceptable for at least the following reasons.

- Neither Alice nor Cullen should not have to accept impersonation, as the guardian will only act on their behalf, not as them.
- The request or court order may cover only a subset of Alice's or Cullen's smart contracts.
- Handing over private keys is not allowed for some types of identity in some jurisdictions[1].
- Alice and Cullen may be unable to hand over their keys, or have plausible deniability[2].

[1] In The Netherlands, citizens have a digital identity (DigID) that enables their interaction with government services. A citizen is not allowed to hand over the keys to its DigID, nor can it hold responsible a third party to whom he handed over the keys.

[2] In The Netherlands, there have been discussions in parliament of a mandatory decryption order, with risk of fines and prisons in case of non-compliance. It was argued that a presumed innocent person

Requirement 5: The solution to requirements 1-4 shall not require the handing over of private keys.

3.2 No mutability of the blockchain

These user stories do not require mutability of the blockchain. The reassigning of authorization over a smart contract is something that could be appended to a blockchain, without needing to alter the past.

Requirement 6: The solution to requirements 1-4 shall not require mutability of the blockchain.

3.3 Backward compatibility

The reassigning of authorization over a smart contract is something that could be implemented by defining one or more new types of transactions. Requirement 1 could be implemented by a new type of transaction that is signed by the original user. Requirement 2 could be implemented by a new type of transaction that is signed by a to-be-defined qualified majority of blockchain-consortium partners, in order to be accepted and appended by the other partners as well. When a blockchain technology does not support this functionality at its genesis, and it is introduced at a later time, then it should preferably also apply to transactions that predate the introduction of this functionality.

Requirement 7: The introduction of a solution to requirements 1-4 should be backward compatible, such that they also apply to smart contracts that predate this introduction.

The requirement is formulated as a “should”. More discussion is needed to understand the technical implications of this requirement, e.g. the amount of generations that allow backwards compatibility.

3.4 Programmability

As blockchains support programmability via smart contracts anyway, it would make sense to enable programmability of after-the-fact mandate changes. Examples of programmability are time dependence (“only between 5PM and 9 AM”, “only until June 10”) or conditional (“only if the XYZ stock exceeds 150 points”). This programmability exceeds the scope of the present document, so it is added as a suggestion.

Requirement 8: The solution to requirements 1-4 may be programmable.

The requirement is formulated as a nice-to-have “may”, as the use case does not require it.

Section 4 - External References and Glossary

should not be sent to prison just for accidentally losing some private keys (plausible deniability). Similar jurisdiction is likely to apply to smart-contract keys.

(Please use this section to add references for standards or well-defined mechanisms.)

- uPort shows a mechanism of reassigning contract ownership [uPort website] [uPort whitepaper
- Decentralized identifiers (DIDs) specify a method for reassigning ownership. The DID spec is still being developed. [did-implementer-draft-10.pdf]
- Permanode - is an anchor from which you build mutable objects [Permanode].