Title

AN INFORMATION SYSTEM THAT AUTOMATES INTERPRETATION AND PERFORMANCE OF CONTRACTS INVOLVING DIGITAL ASSETS

Summary

An information system that automates interpretation and performance of contracts; including functionality for: (1) effecting payments, (2) transferring digital assets, (3) notifying parties; (4) storing of (i) digital assets, (ii) contractual terms, (iii) event history, (iv) event schedule; (5) administering legally binding contractual workflows, including (i) drafting and simulating, (ii) signing, (iii) amendment, and (iv) dispute resolution; and (6) analyzing, summarizing and reporting. Such system is: (1) independent of the parties to the contract; (2) automated; and (3) composes machine interpretable code that forms the actual contractual legal agreement.

Description

1 WHAT IS A SMART CONTRACT?

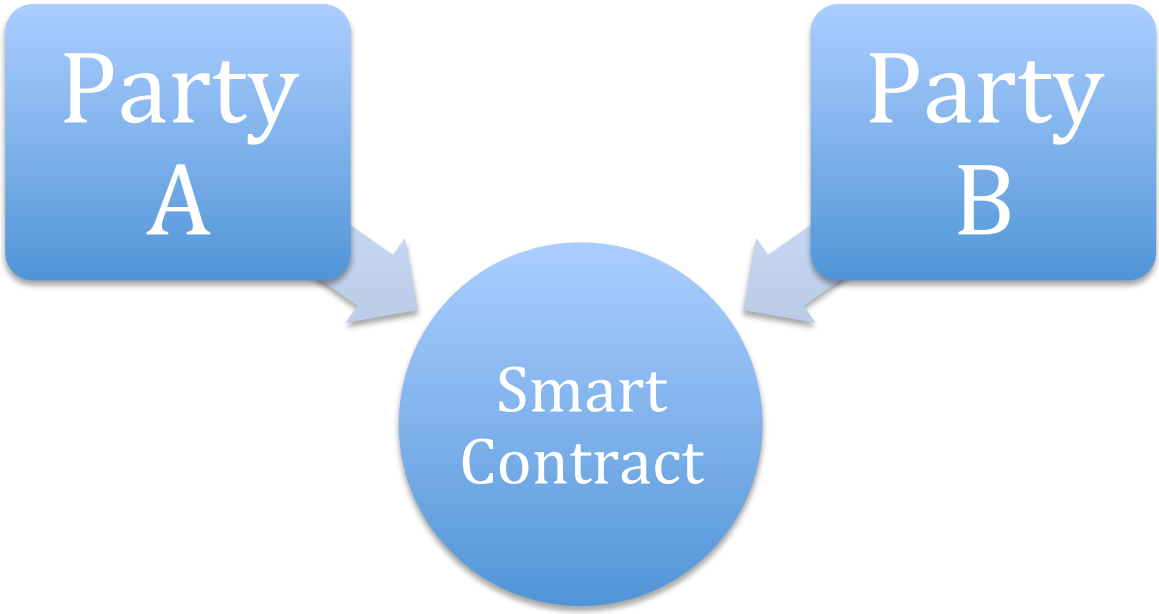
1.1 Overview

The short answer is that Smart Contracts ("SC") are self-executing contracts. SCs are similar to paper contracts written in legalese or natural human language; the major difference is that SCs are written in a computer interpretable language and format. Therefore, it is possible for a computer system to interprets the SC can execute some—if not all—of the terms of the contract.

1.2 Comparison to Existing Workflows

The nearest analogy for a Smart Contract in the traditional view is that of an executor (in the older sense—not its probate definition); or in a more modern sense, an administrator or a servicer of a contract.

An appropriate analogous example is that of two masters that share a servant; the latter has the ability to read and is bound to obey both his masters' signed instructions. When the two masters draft and sign an agreement; the servant, upon receipt of the written contract, would oversee and perform the contractual obligations through to the contract's termination. To draw the parallel back to SCs, the masters are the parties to the contract and the SC is the servant that executes the contract obligations.



1.3 Distinction of Smart Contracts from Digital Contracts

There are various extents of "smarts" for SCs; there are three axes to consider. These aspects are important as to properly distinguish SCs from contracts in digital form. To wit, SCs are a specialization of digital contracts; we must define what aspects of digital contracts that constitute SCs.

For the following analysis, let's assume the following definition:

digital contract	(noun) a digital record that memorializes a contract or agreement between party(ies); usu. a digitized form of a signed contract document.

1.3.1 Level of automation

Defining the spectrum of automation where one end has no automation—i.e. a contract drafted in a way so to not be machine interpretable; and the other end where all terms are machine interpretable.

For a digital contract to be considered a SC, it must have, at a minimum, a non-zero amount of machine interpretable terms.

1.3.2 Separation Between Contract and Execution

A contract could be represented in digital form and completely machine interpretable; however it may not be considered a SC if it is separate from the actual agreement.

Many software systems establish an agreement between the parties via terms of use or a proper written contract; from these terms, the automated system is built according to the

programmer's interpretation—not necessarily the parties' intention.

A SC has its terms established in machine interpretable code; hence avoiding any possible disjunction between the parties' intention and the automated execution.

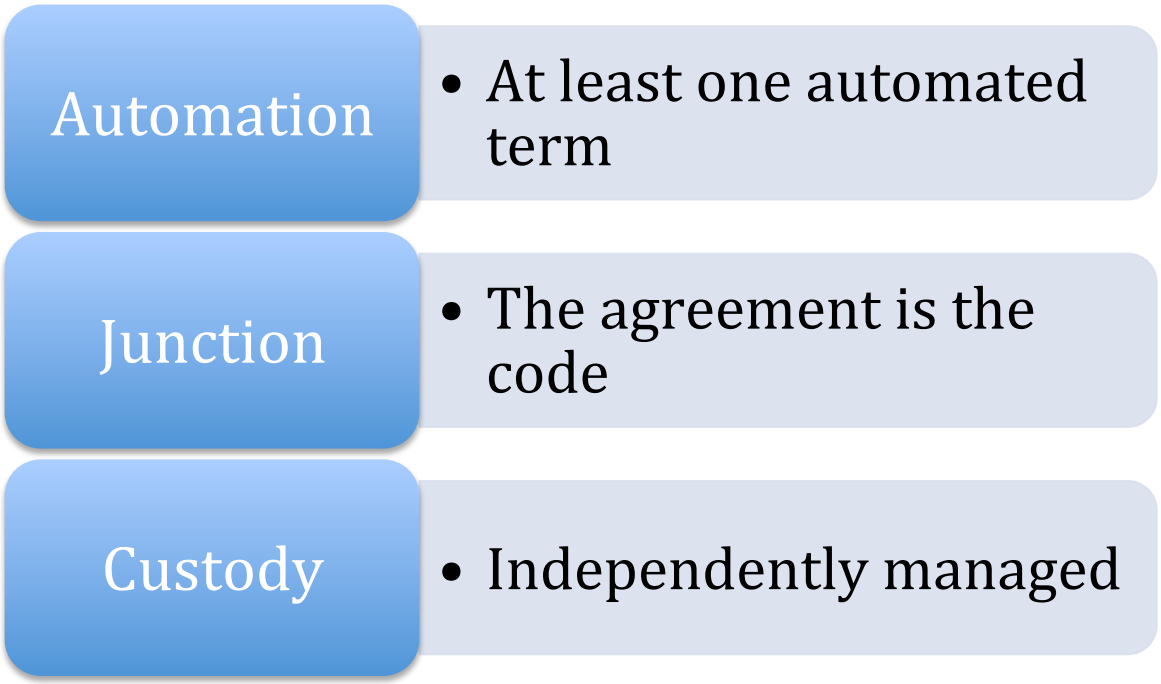
1.3.3 Custody and Oversight

Most software systems are not only drafted as per the programmer's interpretation of the agreement, but are also held and overseen by one party only; giving them free reign over the execution of the terms.

A SC is not kept by one of the parties involved; rather, either by a 3rd party, by both parties, or by a distributed system, e.g. a Blockchain[1].

1.4 Formal Definition

smart contract — (noun) a digital record of a contract or agreement, which (1) contains terms that are machine interpretable and machine executable; (2) represents the actual agreement as accepted by signature; and (3) is stored and administrated by a third party system.



2 Elements of a Smart Contract

Let's examine the elements of a SC.

[1]

2.1 Parties

Although having parties is strictly not necessary for a SC, most cases should have at least 2. Parties are the persons involved in the agreement or transaction represented by the terms of the SC.

2.1.1 Signature and Identification

A party indicates his signature by inserting an encrypted block of data, which should be decryptable in a way that uniquely identifies him. The block of data could be anything, but it ought to be something that makes it clear that it indicates a proper signature—such as the word signature, or the party's name.

To identify the signatory from a signature, we use one or any combination of the following methods:

2.1.1.1 From Public Key Cryptography

Using any public-key cryptography method or standard, one can authenticate another's identity.

2.1.1.1.1 Bi-directional

If the party uses his private key to sign, he is identifiable anytime via his public key, which can decrypt the signature to reveal the signature.

2.1.1.1.2 Uni-directional

If the party uses his public key to sign, he is only identifiable when he appears and produces his private key to decrypt the signature.

2.1.1.2 From Cryptocurrency Wallet

Since a cryptocurrency wallet identifier is unique, it can also be used as means of authentication. It does not act as a signature *per se*, because it cannot by itself identify the party. The reason a signature may not be required is that: (1) the intent to contract is demonstrated if the contract has been affirmed by payment from the wallet in question; and (2) the wallet itself requires the same cryptographic verification as Public Key Cryptography.

2.2 State

Each SC will require state to work its way through its workflows.

2.2.1 Transient

First there is the transient state, which only lasts the duration of one execution pass. This is simply handled by the host computer's memory, within the SCP's allocated memory. This type of memory can only be used for handling immediate processing that can be completed in one pass. The mechanism for this is simple variables in the code, similar to a software program.

2.2.2 Persistent

Secondly, there is the persistent state that lasts for the lifetime of the SC. This requires storage in the SCMP's database. This allows the SC to maintain state throughout its lifetime. The mechanism for this is to have libraries in the SCP that allow CRUD[2] operations on a basic set of values; usually a key-value pair.

2.3 Capacity

Since the SC has its own cryptographic identity, it has the capacity to hold digital assets within its custody.

2.3.1 Cryptocurrency

A SC can hold its own cryptocurrency wallet, as all that is needed is a cryptographic identity. As such, it can both receive and send cryptocurrency payments.

2.3.2 URL

A SC could be provided by the SCMS with its own URI. Provided an active URI, it can both respond to Internet requests (usu. HTTP requests), as well as make requests.

2.3.3 Email Address

A SC could be provided by the SCMS with its own email address with which it can both send and receive email messages.

2.4 Code

This is the operative part of the SC, which contains the instructions of the obligations within the contract.

2.4.1 Events

Events can be defined in the code as functions in code. These functions are callable from external calls; e.g. from a URL request, an email, or a command prompt.

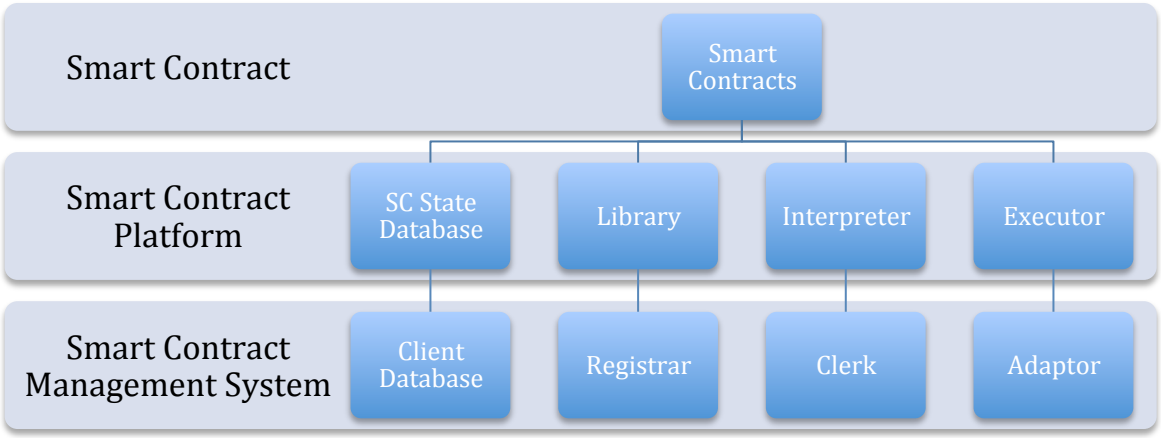
2.4.2 Scope

As a standard software environment, there are multiple scopes of execution: i.e. the global-level and the function- or the event-level.

3 SMART CONTRACT INFRASTRUCTURE

Smart Contracts do not function in the vacuum; they require an infrastructure on which to perform.

[2] Create Read Update Delete



3.1 Smart Contracts

The Smart Contract Level is the actual SC as described in the previous sections.

3.2 Smart Contract Platforms

The Smart Contract Platform ("SCP") is a loosely coupled set of components. Its role is that, given a SC, can interpret and execute it. It is composed of the following components:

3.2.1 Interpreter

The Interpreter component of the SCMS is the module that—as is self-described—interprets the SC's code. When it comes upon an instruction, which requires an external call, it delegates the command to the executor.

3.2.2 Executor

The executor is an optional adaptor between the interpreter and outside systems. This provides an abstraction for tedious instruction routines that may be invoked by simple parameterized calls.

3.2.3 Library

The library is a set of pre-built function calls accessible within the SC's code. They can be categorized as follows.

3.2.3.1 SC Functions

SC functions are functions that perform operations on the SC itself that is being interpreted/executed.

3.2.3.2 External Functions

External functions are functions that perform external operations; e.g. sending email notices. There are covered in more detail in section 4.

3.2.3.3 General Functions

General functions are functions that generally do not constitute performance of obligations and are not covered by other categories; usu. they are related to output and or debugging for aiding development and testing.

3.2.4 SC State Database

The SC State Database is the container of the SC's code, parties and state variables.

3.3 Smart Contract Management Systems

The Smart Contract Management System ("SCMS") is the system that administrates all other aspects of executing SCs, such as: scheduling, persons' profiles, and the contract formation and amendment workflows. It has the following components.

3.3.1 SCPs

The SCPs, as discussed in section 3.2, are responsible for interpreting and executing the SC. A SCMS is ideally loosely coupled to any specific SCP and may make use of one or more SCPs.

3.3.2 Client Database

The Client Database is a database to store client data; e.g. personal data and wallet addresses.

3.3.3 Registrar

The Registrar is the component responsible for administrating the entry and update of client data into the client database by system users; as well as reading of said client data by other system components.

3.3.4 Clerk

The Clerk is the component of the SCMS responsible for submitting SC's code to the Interpreter; which includes scheduling future events, when events are slated to be executed upon a specified date & time.

3.3.5 Adaptor

The Adaptor is an optional component, which can serve to normalize varying behaviors from the SCPs, where external functions are executed. It is functionally a library of functions that can serve as wrapper methods for SCP functions.

4 FUNCTIONS FOR SMART CONTRACT

The SCs inherit some functionality from libraries defined within the SCP and the SCMS. Here is an overview of the functions available.

4.1 HTTP Method Outgoing Calls

Any HTTP based method that is callable on the Internet can be called by the SCMS, on behalf of the SC as it is being executed.

4.1.1 Queries

One type of HTTP method call is that of an HTTP query which returns a value from the remote web server; e.g. an HTTP GET call, which returns a stock price.

4.1.2 Prompts

Another type of HTTP method call is that of an HTTP post that sends a message to a remote web server; e.g. an HTTP POST call, which submits information composing a form.

4.2 HTTP Method Incoming Calls

The other category of HTTP methods are the incoming calls, which are managed by a web server; e.g. as a RESTful web service API.

4.2.1 Basic Reporting

The SCP and SCMS may offer pre-packaged reporting functionality automatically inherited by every SC under it administration.

4.2.2 Custom Reporting

In addition, the SC optionally can define its own reporting functionality.

4.3 Cryptocurrency

The SCMSs and SCPs provide the ability to the SCs to make use of cryptocurrencies in the following ways. To make this possible, the SC has the capacity to have its own cryptocurrency wallet.

Note that this is a specialization of section 4.1.

4.3.1 Balance

The SCs may query the balance of any wallet from the cryptocurrency's Blockchain, given the wallet address; e.g. get the balance of a specified person's Bitcoin wallet.

4.3.2 Transactions

The SCs may query specific cryptocurrency past transactions, to examine its details—i.e. the amount, the sender, the recipient, the date and time sent, the number of Blockchain

confirmations; e.g. confirm the sender of the previous payment made to the SCs wallet.

4.3.3 Automate Payments

The SCs may programmatically trigger a cryptocurrency transaction; e.g. send a Bitcoin payment from its (the SC's own) wallet to another party's wallet.

4.4 Email

The SCMSs and SCPs provide the ability to the SCs to make use of standard email functionality. To make this possible, the SC has the capacity to have its own email account.

4.4.1 Send email

The SCs may send email; e.g. a notice to one of the parties as a reminder to make a payment of a specified amount to a specified party.

4.4.2 Receive email

The SCs may receive email to trigger or record an event; e.g. the SC may have an event triggered when an email arrives to its inbox, given specific keywords another event may be triggered to process an encoded scenario, such as a dispute notice that would freeze the SC's execution.

4.5 SMS

The SCMSs and SCPs provide the ability to the SCs to make use of standardized Short Messaging System ("SMS") functionality. To make this possible, the SC has the capacity to have its own SMS account.

This section's functionality is very similar to section 4.4.

4.5.1 Send message

The SCs may send SMS.

4.5.2 Receive message

The SCs may receive SMS messages to trigger or record an event.

5 NOVELTY

This section discusses where is the novelty?

5.1 Adapted to Legal Systems

Most importantly, the novelty is the adaptation of SCs for Legal Systems, in accordance with Contract Law rules. This makes the SCs legally binding if challenged in public courts.

5.2 Combined Use of Cryptocurrency

Most interesting contracts involve transfers of assets. For SCs to be useful, they require the ability to transact using assets—which were not easily possible before[3]. Bitcoin's Blockchain technology changes this by allowing secure, programmatic asset transfers.

5.3 Automation

Using SCs, one can build and scale large, complex, repeatable transactions, which are to be executed to the letter.

5.4 Security

Cryptocurrencies are inherently secure; furthermore, using similar encryption standards, SCs can be encrypted for authentication and verification of the parties.

5.5 Privacy

Encryption of the SC also allows for privacy of the terms.

5.6 Privity

Given the use of cryptocurrencies, it is very hard for any outside party to intervene into the SC's execution. Furthermore, given the right encryption, the SC can be made unreadable by 3rd parties.

5.7 Transparency

Since the code and the agreement are joined, the actual terms are guaranteed to be transparent to both parties. Additionally, another use of cryptography can guarantee the integrity of the SC via the inclusion of hashes.

5.8 Flexibility

Given privacy and privity of the SC, as well as the use of programming language within a SC development platform, SCs can be drafted in limitless ways—limited only to the parties' imagination (and consent).

5.9 Independence

The parties do not have to have any qualified trust in the other party for the interpretation and automated execution of the SC since it is kept out of control of either party.

6 Smart Contract Management System Methods

The SCMS has an interface from which users or other systems may make use of the functionality.

6.1 Drafting a Contract as a Smart Contract

The SCMS is equipped with a user interface that allows users to draft their own contractual clauses to compose a SC in computer language; similar to email client, or software development environment platform systems.

---

[3] Not due to technological limitations *per se*, but because of strict regulation over banking systems.

6.2 Selecting a Smart Contract Template

The SCMS can list to the user a set of pre-defined templates available for standard contractual purposes. By selecting it, the user can adopt it for its defined purposes.

6.3 Testing and Simulating a Smart Contract

When a SC has code, the SCMS can enter testing and simulating mode, within which the SC's code can be tested and simulated: (1) the SC's state variable are viewable; (2) the user may move forward and backward in time steps; (3) the user may call events; and (4) the user may prompt the system to simulate external events, such as a cryptocurrency payment or an incoming message.

6.4 Proposing a Smart Contract

When a SC has code, the user can specify the parties to the contract, which may or may not include the user himself. Once the required number of parties are specified, the user can prompt the SCMS to 'Propose' that sends a message to each party, which notifies them of there being a SC for their review.

6.5 Reviewing a Smart Contract

When a user receives a notification of a proposed SC, the user may enter the SCMS to view the said SC. The user may view the SC's data, and may test and review it as per section 6.3.

6.6 Counter-Proposing a Smart Contract

When reviewing a SC, as per section 6.5, the user may edit the SC's code. The user may view the SC's data, and may test and review it as per section 6.3. The user may prompt the SCMS to 'Counter-Propose' which sends a message to all other parties, similar to the message described in section 6.4.

Note: in the case where one may follow up on an outdated proposal, the outdated proposal should not persist in the system—only the latest one should be available to review.

6.7 Signing a Smart Contract

When a SC is proposed, it may be signed. The SCMS may be prompted to 'Sign' the SC, which sends a notification to the other parties to the said SC that informs of the said party's signature.

The underlying technical method to effectuate the signature is as described in Section 2.1.1.

6.8 Binding the Parties Without an Express Signature

As a contract can be considered legally binding without a signature, but by conduct in a case of an escrow type contract, where deposits are needed. A deposit from a cryptocurrency wallet can substitute a signature.

Therefore, where all parties require a deposit, no signature is required. All future performance would be either payments to the same cryptocurrency wallet address or to the person as identified by the cryptocurrency wallet.

An example of this is as follows: a contract that encapsulates a sports bet; both parties must deposit the wager amount in order for the bet to take effect—no signature needed; once the final score of match is ascertained, the payment goes to the winner's wallet.

Note: legally speaking, signatures or performance is needed by anyone who incurs liability; but not necessarily by someone who gains only rights from the contract.

6.9 Interpreting Smart Contract Code

Once the SC is coded, established or when it is tested and simulated, its code is interpreted by the SCP. The technical method for a software system to interpret outside code coming from a database requires a development environment that can interpret language without compilation.

The easiest way to achieve this is to build the SCP in a scripting language that can include/require and/or execute/run code from a string or a file.

To achieve this, the SC's code to be interpreted is firstly pulled out of the database, then either put into a temporary file, which is included/required by the scripting language, or the string itself is execute/run.

Note: security measures must be taken so to prevent code-injection attacks.

6.10 Scheduling Smart Contract Events

Events can be scheduled either statically when the contract is being drafted, or programmatically within the SC's code.

Events are represented in the database for persistence. When the SCMS either boots up or when a SC is initialized, or effected, the SCMS registers an Operating System alarm to trigger a call, which subsequently calls the callback method specified by the event registration.

6.11 Notifying a Person

Using any of the functionality as described in Sections 4.1, 4.4 and 4.5, which can be encoded into the SC's code when needed.

6.12 Freezing a Smart Contract

When needed—as required by the SC's code—the SC can be frozen by calling the SCMS's internal call, or by setting the SC's status to 'Frozen.'

6.13 Exporting a Smart Contract's Data

When needed, the SCMS allows the system administrator to pull out all relevant data related to an SC; including code, events, state, values, and parties data.

7 Smart Contract Management System Workflow Methods

A single SC through its lifetime is put through an overarching workflow, which is composed of several sub-workflows, each composing their own pre-conditions.

7.1 Life-Cycle Process

This is the overarching workflow that encompasses most other workflows, and composes the whole life-cycle of the SC, from drafting to completion.

7.1.1 Establishment

The first step to the SC is the drafting workflow, in which the terms are drafted and must be agreed upon by both parties. This workflow is established in section 7.2.

7.1.2 In Effect

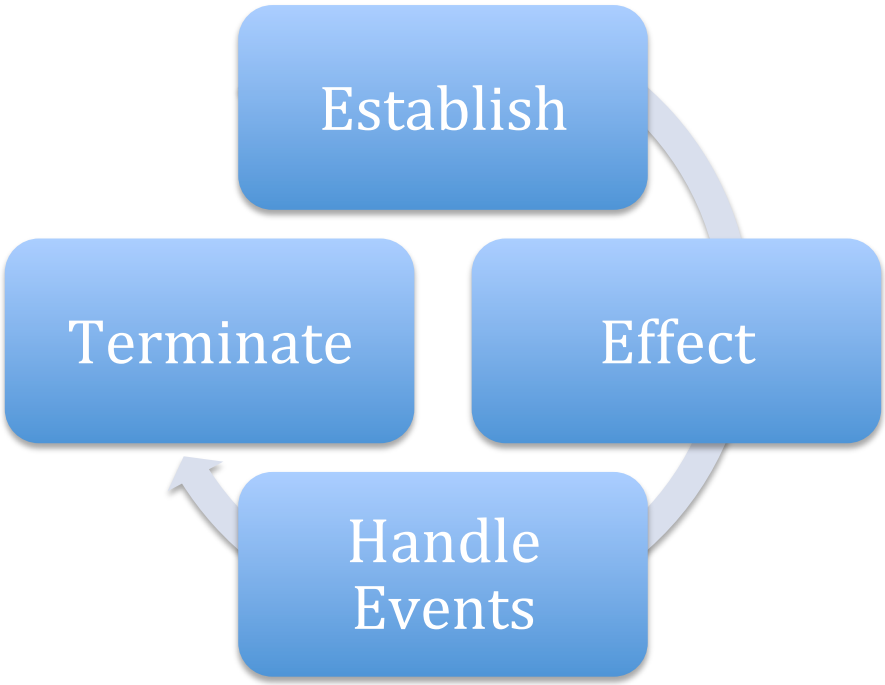
Once the SC is established, the said SC is waiting for events that are either externally triggered, or scheduled.

7.1.3 Events

When an event is called, the specified subsection of code from the SC is interpreted and executed as defined.

7.1.4 Termination

When the contract is deemed to be performed—as per the terms of the SC—, the SC is considered terminated and hence: (1) stops listening to events, and (2) cancels scheduled events; (3) however, it may still respond to general reporting queries.



7.2 Drafting Process

This is the process for drafting and formation of the SC.

7.2.1 Proposal

The first step is for one of the parties to either draft the terms, or get the terms' code from a template.

7.2.1.1 Testing and simulation

The code under the drafting process may be tested and simulated within the SCMS, against different SCPs if needed.

Once the code is considered complete by the original drafting party, he marks it for submission and then the SCMS submits it to the counterparty.

7.2.2 Counter-proposal

Once the counterparty receives the proposal, he may review the SC's terms.

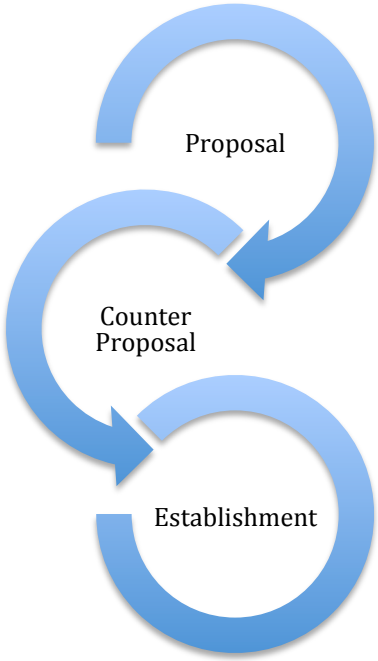
7.2.2.1 Testing and Simulation

As per section 7.2.1.1, the counterparty may optionally test and simulate the SC's code to ensure consistency with his intent.

If the terms are agreeable, the process moves forward; otherwise, he may modify the terms and resubmit the changes for the other party to repeat step 7.2.2.

7.2.3 Establishment

When the SC's terms are agreed upon by both parties, the SC is marked as 'in effect.' The parties are now bound to the terms described within the SC's code.



7.3 Voluntary Amendment Process

This process includes any modification to the terms on which the parties have agreement; including (1) general amendment, (2) early termination, (3) frustration, and (4) mediation.

7.3.1 Proposal

The first step is for one of the parties to either draft the modified terms. A duplicate SC is created on which the party may make the changes.

Note: the existing SC is not frozen and is unaffected by the drafting of an amendment, unless the SC is indicated otherwise in its terms.

7.3.1.1 Testing and simulation

The code may be tested and simulated within the SCMS, against different SCPs if needed.

Once the code is considered complete by the original drafting party, he marks it for submission and then the SCMS submits it to the counterparty.

7.3.2 Counter-proposal

Once the counterparty receives the amendment proposal, he may review the SC's changes, which are identified from the original content.

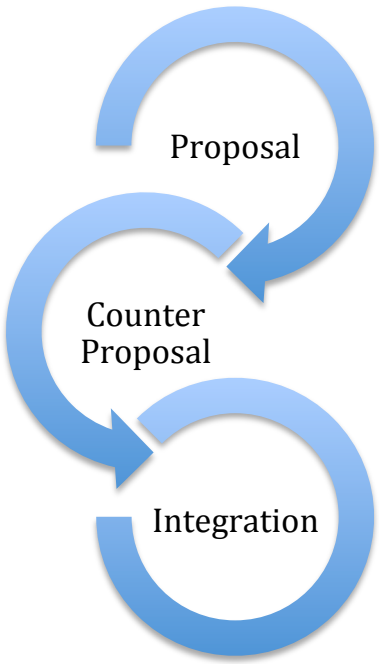
7.3.2.1 Testing and Simulation

As per section 7.3.1.1, the counterparty may optionally test and simulate the SC's code to ensure consistency with his intent.

If the terms are agreeable, the process moves forward; otherwise, he may modify the terms and resubmit the changes for the other party to repeat step 7.3.2.

7.3.3 Integration

Once the parties agree on the terms of the amendment, the integration process is kicked off by the SCMS. This step involves: (1) freezing the old SC; (2) effecting the new SC; (3) porting the relevant event history from the old SC to the new SC.



7.4 Internal Arbitration Process

SCs allow the inclusion of encoded terms, which may govern an internal arbitration process.

7.4.1 Preconditions

The preconditions for this process, as discussed above, is that the workflow must be encoded as part of the SC's terms, which could be standardized terms provided by the SCMS. All of section 7.4 moving forward assumes that this precondition is satisfied.

The required terms are prescribed by law, but are generally as follows: (1) appointment of arbitrator, (2) seat of the agreement, and (3) jurisdiction of the governing law.

7.4.2 Notice of Dispute

While a SC is in effect, any party—unless otherwise specified in the terms—may raise a dispute by submitting a properly formatted notice of dispute. Depending on the agreed upon terms, the SC may be frozen, preventing any execution from being processed.

Additionally, as prescribed by law, the party raising the dispute must submit a claim.

Optionally, the parties may privately resolve the dispute as per section 7.3, in which case this process does not progress.

7.4.3 Hearing

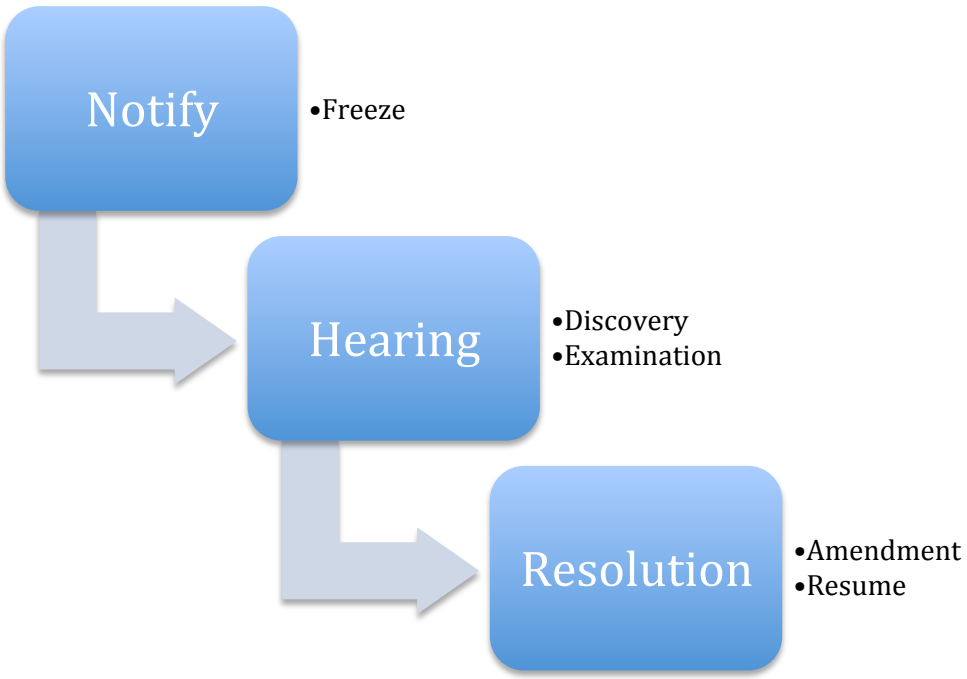
Once the notice of dispute has been served and the claim submitted, the appointed arbitrator has the opportunity to evaluate the claim.

The arbitrator's keys must firstly be registered into the system and the SC, so that the arbitrator may export all of the SC's data, as defined in 3.2.4. Ideally, the arbitrator's keys ought to have been registered during the drafting process of section 7.2, otherwise the SCMS may unilaterally register them at this stage.

Once the arbitrator has discovery of all of the SC's data, he may examine the data in consideration of the disputed claim. The arbitrator then proceeds through the arbitration process, which possibly includes hearings, etc, as prescribed by law and otherwise as per the arbitrator's discretion.

7.4.4 Resolution

Once the arbitration process is completed: in the case of a dismissal, the SC's execution resumes by repealing its frozen state; in the case of an award, the arbitrator unilaterally amends the SC's code to effectuate the award.



7.5 External Dispute

As opposed to section 7.4, this workflow applies when an outside body assumes

jurisdiction in resolving a dispute; such as a court.

In this scenario, the SCMS, assuming that it is under the court's jurisdiction, will be bound to comply to court orders.

7.5.1 Injunction

In the case of a court order for an injunction of the SC, the SCMS will: (1) freeze the SC's execution; and (2) export the SC's data as per section 3.2.4.

7.5.2 Resolution

Once the court case is ruled upon, it is up to the parties involved to notice the SCMS with the judgment to effectuate the resolution: either (1) to resume the SC's operations; (2) to terminate the SC; or (3) to unilaterally amend the terms according to the judgment.

Claims

1. An information system that includes digital storage of information relating to one or more persons and their contractual agreements.
    a. That said contractual agreements' clauses, as in Claim 1, can be written in either: (1) natural language, or (ii) machine interpretable language.
    b. That said personal data includes said persons', as in Claim 1: (i) personally identifying information, (ii) cryptography credentials, and/or (iii) cryptocurrency wallet addresses.
    c. That said contract agreements' data, as in Claim 1, composes its: (i) historical data, (ii) event schedule, (iii) state variables, and/or (iv) other data.
2. An information system as in Claim 1 that performs contractual obligations by executing the terms of the contracts as written in machine interpretable language.
    a. That contracts in digital format in said system, as in Claim 2, each have a unique cryptographic identity.
    b. That contracts in digital format in said system, as in Claim 2, each have a unique cryptographic identity in order to possess cryptocurrency wallets.
    c. That contracts in digital format in said system, as in Claim 2, each have a unique cryptographic identity in order to possess digital assets.
    d. That said system, as in Claim 2, schedules events for future contractual tasks of its contracts in digital formats.
    e. That said system, as in Claim 2, responds to Internet queries and posts in performance of a contract in digital format.
    f. That said system, as in Claim 2, responds to messages in any electronic format in performance of a contract in digital format.
    g. That said system, as in Claim 2, sends Internet messages in performance of a contract in digital format.
    h. That said system, as in Claim 2, sends messages in any electronic format in performance of a contract in digital format.
3. A method to draft contracts in digital format that are written in machine interpretable language for use in the said information system, as in Claim 2.
4. A method to test and/or simulate contracts in digital format that are written in machine interpretable language for use in the said information system, as in Claim 2.
5. A method to propose contracts in digital format that are written in machine interpretable language for use in the said information system, as in Claim 2.
6. A method to review proposed contracts in digital format that are written in machine interpretable language for use in the said information system, as in Claim 2.
7. A method to counter-propose contracts in digital format that are written in machine interpretable language for use in the said information system, as in Claim 2.
8. A method to sign contracts in digital format for use in the said information system, as in Claim 2.

9. A method for binding parties to contracts in digital format without the need for a signature.
10. A method for the said information system, as in Claim 2, to interpret contracts in digital format that are written in machine interpretable language.
11. A method for the said information system, as in Claim 2, to schedule events on which contractual obligations are to be performed.
12. A method for the said information system, as in Claim 2, to notify either: (i) a party to the contract in digital format, or (ii) a third party.
13. A method for the said information system, as in Claim 2, in order to manage a contract in digital format throughout its life cycle from draft to completion.
14. A method for the said information system, as in Claim 2, in order to establish a contract in digital format.
15. A method for the said information system, as in Claim 2, in order to amend a contract in digital format.
    a. A method for voluntary termination based on Claim 15.
    b. A method for integrating a mediation award based on Claim 15.
16. A method for freezing the contract's performance within the said information system, as in Claim 2,
17. A method for export a relevant contract's data from the said information system, as in Claim 2,
18. A method for unilaterally amend the contract's terms within the said information system, as in Claim 2,