Explainer | Smart Contracts

Jump to:

Quick Links:

What are Smart Contracts?

To begin with, smart contracts are neither particularly smart nor are they, strictly speaking, contracts.

That aside, they do provide a fairly well placed metaphor for what they really are: blockchain (https://monax.io/explainers/blockchains) housed scripts which represent unilateral promises to provide a determinate computation based on transactions which are sent to the script.

These scripts are compiled into low level operation codes and stored in the blockchain’s data store at a particular address – which is determined when the contracts are deployed to the blockchain. When a transaction is sent to that address the distributed virtual machine on every full node of the blockchain network executes the script’s operation codes using the data which is sent with the transaction.

Smart contracts are modular, repeatable, autonomous scripts, which can be used to build applications for yourself, for a community, for a client, for a bounty, or even just for fun. They can be mixed and matched, and easy to iterate, rather like lego bricks combined with pre-set templates.

Smart contracts can be coded to reflect any kind of business or engineering logic which is data-driven: from actions as simple as up-voting a post on a forum, to the more complex such as loan collateralisation and futures contracts, to the highly complex such as repayment prioritisation on a structured note.

Relationships and obligations which are ‘smart contractified’ benefit from blockchain security logic and also the increase in verifiability that blockchain networks provide. They also have the benefit of being operable in the exact same manner across stakeholders involved in a particular deal or application.

By building business logic in smart contracts, developers (and lawyers) can give their users and clients an increase in the verifiability and certainty which comes with distributed technology while simultaneously building a system of rules which will be structured so that it can keep up with increases in automation in the world around us.

Benefits of Using Smart Contracts

There are many benefits of using smart contracts; not the least of which is a large increase in verifiability of our data-driven relationships and their outcomes or knock-on effects.

While, at their core, smart contracts are simply software scripts no different than those which may run in any application stack, they have one unique quality to them which any other random software script does not: certainty.

Smart contracts have visibility across the blockchain they live on. That is to say, if someone has access to read the blockchain, they will have access to see the compiled script. This is a very different idea than being told that a certain script will operate in a certain way on servers which you may not control.

Let us step back for a moment from the specifics of smart contracts and think about a commercial deal which will involve a dozen entities and many different data-driven interactions throughout the life of the deal. How the IT departments for the commercial entities would likely structure the administration of the data driven aspects of the deal would be to establish a tracking system which is completely under the control of that commercial entity and, perhaps, provide some programmatic interface for others to query records or send new information regarding a record. It is likely that each of these dozen commercial entities will each establish a similar system to monitor and track the data interactions over the course of the deal.

This is where the execution certainty of smart contracts married with the historical transaction certainty of a blockchain should become increasingly interesting for commercial players. If the commercial entities were wise about how they structured the deal, they could track all of their data-driven interactions on a smart contract-enabled blockchain without having to build twelve different systems, ensure their interoperability, and expend labour-time to appropriately categorize and file relevant transactional data. Every entity having access to the blockchain will be able to completely verify the entirety of the interactions as well as the entire history of the data set, which would be automatically maintained over the life of the deal and summarised at its conclusion.

Under the current software design paradigm which would likely be deployed for such a deal, this would never happen. Each entity would keep full control over the scripts or software tracking and executing any downstream functions after a new or update record transaction.

While the different commercial entities may be legally incorporated and would not be required to move to a completely trustless system of the kind described above, they would all experience significant cost savings from the increased verifiability and automation a blockchain-based smart contract system would permit.

Challenges of Working With Smart Contracts

At the moment, there are two primary limitations to working with smart contracts. The first challenge is temporary, the second challenge is fundamental.

The first challenge of working with smart contracts is that there is something of a prisoner's game with respect to their adoption. That is to say: the most significant benefits of smart contract adoption come when numerous commercial entities begin to automate their data-driven interactions using smart contracts and a blockchain which is purpose-built for that (and preferably only that) multi-party interaction.

In a single commercial entity's context, in the short-term the design and deployment of smart contract systems would for many applications be less efficient than carrying on with a centralized software stack with redundancy built-in.

However, in the context of multiple commercial entities, the redundancies of data and scripting capabilities required for multiple parties to track and manage a deal tips the efficiency scales heavily toward smart contracts, with marginal returns increasing over time with subsequent iterations and increasing blockchain use throughout a given market.

The second challenge of working with smart contracts is that smart contracts are simply software. They are not the "living, breathing" documents that people generally think about when they think of "contracts" and as such they are not inherently enforceable.

Indeed, they can "enforce" or, better, administer some of the state of the data which they have access to on the blockchain on which they reside. Yet, beyond that there is little reach that smart contracts have. For the foreseeable future they will not be enforceable in any court, and few parties will be able to rely on smart contract technology alone to structure all of the terms of a commercial transaction in code.

Comparing Smart Contracts to "Normal Scripts"

Are smart contracts better than a simple python script?

This is a question which in the early days of Monax we fielded often. This is a debatable subject. And answering it turns on two axes.

Axis 1 => spectrum of verifiability

A python script is *somewhat* verifiable. If a python script is running on somebody else's metal your ability to verify what its doing is usually limited to observing its results. You can always see what the script is doing if, for example it is managing an API. You know that you send that API a request and then you will get a response. But what is happening between the request and the response is very difficult to verify.

Even if you could verify *the code that was being ran* via some fingerprinting mechanism you still wouldn't necessarily be able to verify *the execution environment* of that script.

While it is true that it is a relatively straight-forward problem to verify a script being run *on your machines* is the same script which was produced by Skype, Microsoft, or whomever; it is not necessarily a simple problem to verify that a script being run *on someone else's machine* is the same script produced.

Another problem is that scripts can read top level environment variables, and run differently on a different version of their language; therefore they can have differing **outcomes** depending on what the environment variables of the Operating System are or what version of the language is "running" (or compiling) the language.

What this means is that when a computer script is ran it can be told by the operating system "how" it should run in some instances and in other instances various versions of a language will operate differently. As a result of this script makers have an ability to make their scripts run differently on different versions of the language.

The execution environment in which the script runs is important because the same script can run different ways depending on its environment. This is important to understand because even if you can verify that the script a machine is running is the same script that was produced by a script maker, what you are *really* trying to verify is the predictability of

what its **doing** (not only what its numbers, meaning its code, is).

The practical upshot here is that nobody really has a capability to verify code that's running on someone else's metal.

And this is one of the most powerful capabilities which smart contract backed systems offer to their users. Smart contracts completely isolate the logic and data into a "casing" (provided by a blockchain) which is utterly verifiable. Every compute step along the logic sequence is verified by every node on the network.

Those nodes could be other banks within a consortium, internal audit, external audit, the business's accounting department, your grandmother, or whomever is in the network. But all of these nodes will be checking each other's work.

Now this is overkill for many, many computing requirements which an enterprise may have (indeed the **vast** majority of an enterprise's computing requirements do not need this level of computation verifiability).

But for instances where one has a **data driven relationship** – whether that is a compliance relationship with a regulator, a customer relationship, or a peer relationship – it may be a price which institutions are willing to pay. In some contexts.

But. And this is the key. It is `certainly` very different than a simple python script running on someone else's metal.

Axis 2 => spectrum of privacy

While a python script is not highly verifiable, it **is** easy to hide and to keep protected. It's been said over and over again, blockchains are transparency machines, they are not privacy machines. There currently is no blockchain backed smart contract mechanism which can provide the level of verifiability described above **and** that can also provide *any* level of privacy beyond simple obscuration of the data on the blockchain.

If you need privacy use scripts. Not smart contracts.

Are Smart Contracts Legal Tech?

*Simply put, smart contracts provide the backbone for automating business processes which reach *outside* of the rotating glass doors.*

One of the ways in which these `scripts` are, kind of, sort of, `contracts` (at least they can be programmed to fulfill many of the legal requirements of real world commercial contracts) is that counter-parties to an agreement have never had an ability to automate their data driven *relationships* via a mechanism which they did not wholly control.

Great strides have been made in industry around automation of a broad range of business *processes*, but the range of business processes which has been automated for the most part stops at the rotating glass door.

In other words, if you're the big player and you can run the computation sequence on your metal then great, but everybody else either has to duplicate the work of formulating the automation and process management components on *their side* of the data driven relationship – or they have to just trust you.

Smart contracts offer a third way, a new paradigm, wherein legally binding agreements backed up by real world agreements can be built to run within a network of computers which no single party can pull the plug on and in which all parties to the agreement **participate** in the management and supervision of the computers which have automated the agreement.

In other words, the sweet spot for smart contracts is data driven *relationships* – or business processes across stakeholders.

To get started programming Smart Contracts, please see our Solidity Series on Smart Systems of Smart Contracts (https://monax.io/docs/solidity/).

To learn more about the differences in approach of a tokenized blockchain approach *vis a vis* a smart contract approach (https://monax.io/explainers/contracts_v_tokens/) please see our explainer on the subject.

All Explainers (https://monax.io/explainers/)