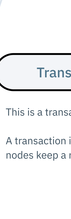
Cerberus - How Radix achieves infinite linear scalability while preserving atomic composability

Substate and Transactions

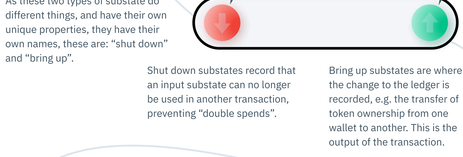
So if you think back to how Bitcoin works, transactions are formed out of Unspent Transaction Outputs (UTXOs).



The Bitcoin transaction to the left consists of three UTXOs, where each one is "spending" the output from a previous Bitcoin transaction. Once spent, a UTXO cannot be used again in another transaction. You can trace any fraction of a Bitcoin through a sequence of UTXOs all the way back to when it was first "mined".

Radix takes a similar approach, but instead of just simple token transfers such as in Bitcoin, substate can represent anything - such as smart contracts.

To make a change to the ledger, a transaction must take one or more substates as input, shut down those substate(s), and then bring up one or more substate(s) as output.



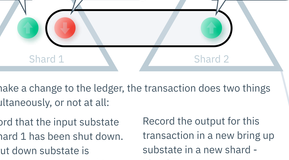
This is a transaction in Radix.

A transaction is a unit of data that nodes keep a record of.

Transactions contain substates. There's a minimum of two types of substate that must be created for a transaction to work. On the left, a substate to record that the substate from a previous transaction, the "input" substate, has been shut down. On the right, an "output" substate to bring up a change to the ledger.

If the transaction is complex, it could involve many different substates of either type.

As these two types of substate do different things, and have their own unique properties, they have their own names, these are: "shut down" and "bring up".



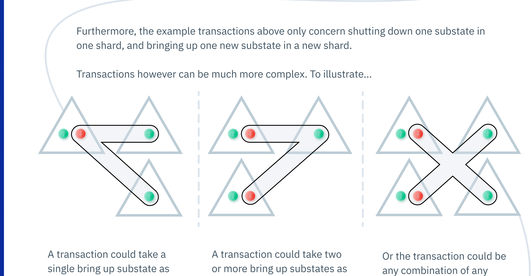
Shut down substates record that an input substate can no longer be used in another transaction, preventing "double spends".

Bring up substates are where the change to the ledger is recorded, e.g. the transfer of token ownership from one wallet to another. This is the output of the transaction.

So at the beginning of the shardspace and validator sets episode, we mentioned that every substate has its own shard. However, to be more precise, it is every bring up substate that has its own shard; and when that bring up substate is shut down, it is joined by a corresponding shut down substate in that same shard. A shard will thus over its lifecycle:

- begin empty
- contain a bring up substate
- contain a bring up substate and a corresponding shut down substate.

To illustrate, we walk through an example transaction below.



This shows Shard 1, with a bring up substate within it.

This is the output substate from a previous transaction. However, this will be the input substate for our example transaction to the right.

As this bring up substate has not yet been shut down, it has the shard to itself.

To make a change to the ledger, the transaction does two things simultaneously, or not at all:

Record that the input substate in Shard 1 has been shut down. A shut down substate is created in the same shard to record this.

The transaction links the shut down substate in Shard 1 with the bring up substate in Shard 2. This is so that anyone looking at the output can see the related input that made it possible.

The substate in Shard 1 can never be used again in another transaction, as there is now a shut down substate in that shard.

Record the output for this transaction in a new bring up substate in a new shard - Shard 2.

The new bring up substate is linked back to Shard 1 by the transaction, immutably tying the two shards together – for that transaction only. Anyone who looks at either shard can thus follow the transaction in either direction.

At some point in the future, this bring up substate will itself be shut down, and the cycle continues...

...forming a chain of bring up and shut down substates across many different shards, across many different transactions through time. These substates are connected cryptographically as inputs and outputs – similar to a blockchain, but on a much more individual scale rather than a single global ledger chain of blocks. You can follow any substate on the ledger all the way back to the genesis of the Radix ledger.

Furthermore, the example transactions above only concern shutting down one substate in one shard, and bringing up one new substate in a new shard.

Transactions however can be much more complex. To illustrate...

A transaction could take a single bring up substate as input, shutting it down to create two or more bring up substates as output.

A transaction could take two or more bring up substates as input, shutting both of them down to create a single bring up substate as output.

Or the transaction could be any combination of any input or output substates, limited only by computing power and network capacity.

As a final thought, it's worth emphasizing that all transactions are cross-shard - it's not possible to have a transaction within a single shard.

The transaction thus serves as a wrapper, binding all the substates inside a single operation, so that they must be accepted either all together, or not at all. This is what "atomic" means.

So how are substates allocated shards?

The version of Cerberus described in this infographic series is scheduled to launch as part of the fully sharded Radix Xi'an release. Please visit www.radixdlt.com for details on the Radix roadmap.

© Copyright Radix Tokens (Jersey) Limited