Object Model

The basic unit of storage in Sui is the object. In contrast to many other blockchains where storage is centered around accounts containing key-value stores, Sui's storage is centered around objects addressable on-chain by unique IDs. A smart contract is an object (called a Sui Move package), and these smart contracts manipulate objects on the Sui network:

Each Sui object has the following metadata:

In addition to common metadata, objects have a category-specific, variable-sized contents field containing a Binary Canonical Serialization (BCS) -encoded payload.

There are a few different ways to concisely refer to an object without specifying its entire contents and metadata, each with slightly different use cases:

Transactions take objects as input, read/write/mutate these inputs, and produce mutated or freshly created objects as output. Each object knows the (hash of the) last transaction that produced it as an output. Thus, a natural way to represent the relationship between objects and transactions is a directed acyclic graph (DAG) where:

The root of this DAG is a genesis transaction that takes no inputs and produces the objects that exist in the system's initial state. The DAG can be extended by identifying mutable transaction outputs that have not yet been consumed by any committed transaction and sending a new transaction that takes these outputs (and optionally, immutable transaction outputs) as inputs.

The set of objects that are available to be taken as input by a transaction are the live objects, and the global state maintained by Sui consists of the totality of such objects. The live objects for a particular Sui address A are all objects owned by A, along with all shared and immutable objects in the system.

When this DAG contains all committed transactions in the system, it forms a complete (and cryptographically auditable) view of the system's state and history. In addition, you can use the scheme above to construct a DAG of the relevant history for a subset of transactions or objects (for example, the objects owned by a single address).

Sui has some limits on transactions and data used in transactions, such as a maximum size and number of objects used. For more information on limits, see Building against Limits in The Move Book.

The ProtocolConfig struct in the sui-protocol-config crate itemizes these limits. Expand the following code to see the ProtocolConfig struct and the comments that explain each parameter.

lib.rs

Select a network from the following tabs to see the currently configured limits and values.

Object metadata

Each Sui object has the following metadata:

In addition to common metadata, objects have a category-specific, variable-sized contents field containing a Binary Canonical Serialization (BCS) -encoded payload.

There are a few different ways to concisely refer to an object without specifying its entire contents and metadata, each with slightly different use cases:

Transactions take objects as input, read/write/mutate these inputs, and produce mutated or freshly created objects as output. Each object knows the (hash of the) last transaction that produced it as an output. Thus, a natural way to represent the relationship between objects and transactions is a directed acyclic graph (DAG) where:

The root of this DAG is a genesis transaction that takes no inputs and produces the objects that exist in the system's initial state. The DAG can be extended by identifying mutable transaction outputs that have not yet been consumed by any committed transaction and sending a new transaction that takes these outputs (and optionally, immutable transaction outputs) as inputs.

The set of objects that are available to be taken as input by a transaction are the live objects, and the global state maintained by Sui consists of the totality of such objects. The live objects for a particular Sui address A are all objects owned by A, along with all shared and immutable objects in the system.

When this DAG contains all committed transactions in the system, it forms a complete (and cryptographically auditable) view of the

system's state and history. In addition, you can use the scheme above to construct a DAG of the relevant history for a subset of transactions or objects (for example, the objects owned by a single address).

Sui has some limits on transactions and data used in transactions, such as a maximum size and number of objects used. For more information on limits, see Building against Limits in The Move Book.

The ProtocolConfig struct in the sui-protocol-config crate itemizes these limits. Expand the following code to see the ProtocolConfig struct and the comments that explain each parameter.

lib.rs

Select a network from the following tabs to see the currently configured limits and values.

Referring to objects

There are a few different ways to concisely refer to an object without specifying its entire contents and metadata, each with slightly different use cases:

Transactions take objects as input, read/write/mutate these inputs, and produce mutated or freshly created objects as output. Each object knows the (hash of the) last transaction that produced it as an output. Thus, a natural way to represent the relationship between objects and transactions is a directed acyclic graph (DAG) where:

The root of this DAG is a genesis transaction that takes no inputs and produces the objects that exist in the system's initial state. The DAG can be extended by identifying mutable transaction outputs that have not yet been consumed by any committed transaction and sending a new transaction that takes these outputs (and optionally, immutable transaction outputs) as inputs.

The set of objects that are available to be taken as input by a transaction are the live objects, and the global state maintained by Sui consists of the totality of such objects. The live objects for a particular Sui address A are all objects owned by A, along with all shared and immutable objects in the system.

When this DAG contains all committed transactions in the system, it forms a complete (and cryptographically auditable) view of the system's state and history. In addition, you can use the scheme above to construct a DAG of the relevant history for a subset of transactions or objects (for example, the objects owned by a single address).

Sui has some limits on transactions and data used in transactions, such as a maximum size and number of objects used. For more information on limits, see Building against Limits in The Move Book.

The ProtocolConfig struct in the sui-protocol-config crate itemizes these limits. Expand the following code to see the ProtocolConfig struct and the comments that explain each parameter.

lib.rs

Select a network from the following tabs to see the currently configured limits and values.

The transaction-object DAG: Relating objects and transactions

Transactions take objects as input, read/write/mutate these inputs, and produce mutated or freshly created objects as output. Each object knows the (hash of the) last transaction that produced it as an output. Thus, a natural way to represent the relationship between objects and transactions is a directed acyclic graph (DAG) where:

The root of this DAG is a genesis transaction that takes no inputs and produces the objects that exist in the system's initial state. The DAG can be extended by identifying mutable transaction outputs that have not yet been consumed by any committed transaction and sending a new transaction that takes these outputs (and optionally, immutable transaction outputs) as inputs.

The set of objects that are available to be taken as input by a transaction are the live objects, and the global state maintained by Sui consists of the totality of such objects. The live objects for a particular Sui address A are all objects owned by A, along with all shared and immutable objects in the system.

When this DAG contains all committed transactions in the system, it forms a complete (and cryptographically auditable) view of the system's state and history. In addition, you can use the scheme above to construct a DAG of the relevant history for a subset of transactions or objects (for example, the objects owned by a single address).

Sui has some limits on transactions and data used in transactions, such as a maximum size and number of objects used. For more

information on limits, see [Building against Limits]() in The Move Book.

The ProtocolConfig struct in the [sui-protocol-config]() crate itemizes these limits. Expand the following code to see the ProtocolConfig struct and the comments that explain each parameter.

lib.rs

Select a network from the following tabs to see the currently configured limits and values.

Limits on transactions, objects, and data

Sui has some limits on transactions and data used in transactions, such as a maximum size and number of objects used. For more information on limits, see [Building against Limits]() in The Move Book.

The ProtocolConfig struct in the [sui-protocol-config]() crate itemizes these limits. Expand the following code to see the ProtocolConfig struct and the comments that explain each parameter.

lib.rs

Select a network from the following tabs to see the currently configured limits and values.

Related links