The Move Book

Every transaction has the execution context. The context is a set of predefined variables that are available to the program during execution. For example, every transaction has a sender address, and the transaction context contains a variable that holds the sender address.

The transaction context is available to the program through the TxContext struct. The struct is defined in the sui::tx_context module and contains the following fields:

Transaction context cannot be constructed manually or directly modified. It is created by the system and passed to the function as a reference in a transaction. Any function called in a Transaction has access to the context and can pass it into the nested calls.

TxContext has to be the last argument in the function signature.

With only exception of the ids_created , all of the fields in the TxContext have getters. The getters are defined in the sui::tx_context module and are available to the program. The getters don't require &mut because they don't modify the context.

The TxContext is required to create new objects (or just UID s) in the system. New UIDs are derived from the transaction digest, and for the digest to be unique, there needs to be a changing parameter. Sui uses the ids_created field for that. Every time a new UID is created, the ids_created field is incremented by one. This way, the digest is always unique.

Internally, it is represented as the derive_id function:

The underlying derive_id function can also be utilized in your program to generate unique addresses. The function itself is not exposed, but a wrapper function fresh_object_address is available in the sui::tx_context module. It may be useful if you need to generate a unique identifier in your program.

Reading the Transaction Context

With only exception of the ids_created , all of the fields in the TxContext have getters. The getters are defined in the sui::tx_context module and are available to the program. The getters don't require &mut because they don't modify the context.

```
bash public fun some_action(ctx: &TxContext) { let me = ctx.sender(); let epoch = ctx.epoch(); let digest = ctx.digest(); // ... }
```

The TxContext is required to create new objects (or just UID s) in the system. New UIDs are derived from the transaction digest, and for the digest to be unique, there needs to be a changing parameter. Sui uses the ids_created field for that. Every time a new UID is created, the ids_created field is incremented by one. This way, the digest is always unique.

Internally, it is represented as the derive_id function:

```
bash // File: sui-framework/sources/tx_context.move native fun derive_id(tx_hash: vector<u8>, ids_created: u64): address;
```

The underlying derive_id function can also be utilized in your program to generate unique addresses. The function itself is not exposed, but a wrapper function fresh_object_address is available in the sui::tx_context module. It may be useful if you need to generate a unique identifier in your program.

```
bash // File: sui-framework/sources/tx_context.move /// Create an `address` that has not been used. As it is an object address, it will never /// occur as the address for a user. /// In other words, the generated address is a globally unique object ID. public fun fresh_object_address(ctx: &mut TxContext): address { let ids_created = ctx.ids_created; let id = derive_id(*&ctx.tx_hash, ids_created); ctx.ids_created = ids_created + 1; id }
```

Mutability

The TxContext is required to create new objects (or just UID s) in the system. New UIDs are derived from the transaction digest, and for the digest to be unique, there needs to be a changing parameter. Sui uses the ids_created field for that. Every time a new UID is created, the ids_created field is incremented by one. This way, the digest is always unique.

Internally, it is represented as the derive_id function:

```
bash // File: sui-framework/sources/tx_context.move native fun derive_id(tx_hash: vector<u8>, ids_created: u64): address;
```

The underlying derive_id function can also be utilized in your program to generate unique addresses. The function itself is not exposed, but a wrapper function fresh_object_address is available in the sui::tx_context module. It may be useful if you need to generate a unique identifier in your program.

```
bash // File: sui-framework/sources/tx_context.move /// Create an `address` that has not been used. As it is an object address, it will never /// occur as the address for a user. /// In other words, the generated address is a globally unique object ID. public fun fresh_object_address(ctx: &mut TxContext): address { let ids_created = ctx.ids_created; let id = derive_id(*&ctx.tx_hash, ids_created); ctx.ids_created = ids_created + 1; id }
```

Generating unique addresses

The underlying derive_id function can also be utilized in your program to generate unique addresses. The function itself is not exposed, but a wrapper function fresh_object_address is available in the sui::tx_context module. It may be useful if you need to generate a unique identifier in your program.

```
bash // File: sui-framework/sources/tx_context.move /// Create an `address` that has not been used. As it is an object address, it will never /// occur as the address for a user. /// In other words, the generated address is a globally unique object ID. public fun fresh_object_address(ctx: &mut TxContext): address { let ids_created = ctx.ids_created; let id = derive_id(*&ctx.tx_hash, ids_created); ctx.ids_created = ids_created + 1; id }
```