The Move Book

Sui has two ways of accessing the current time: Epoch and Time . The former represents operational periods in the system and changed roughly every 24 hours. The latter represents the current time in milliseconds since the Unix Epoch. Both can be accessed freely in the program.

Epochs are used to separate the system into operational periods. During an epoch the validator set is fixed, however, at the epoch boundary, the validator set can be changed. Epochs play a crucial role in the consensus algorithm and are used to determine the current validator set. They are also used as measurement in the staking mechanism.

Epoch can be read from the transaction context :

It is also possible to get the unix timestamp of the epoch start:

Normally, epochs are used in staking and system operations, however, in custom scenarios they can be used to emulate 24h periods. They are critical if an application relies on the staking logic or needs to know the current validator set.

For a more precise time measurement, Sui provides the Clock object. It is a system object that is updated during checkpoints by the system, which stores the current time in milliseconds since the Unix Epoch. The Clock object is defined in the sui::clock module and has a reserved address 0x6 .

Clock is a shared object, but a transaction attempting to access it mutably will fail. This limitation allows parallel access to the Clock object, which is important for maintaining performance.

There is only one public function available in the Clock module - timestamp_ms . It returns the current time in milliseconds since the Unix Epoch.

Epoch

Epochs are used to separate the system into operational periods. During an epoch the validator set is fixed, however, at the epoch boundary, the validator set can be changed. Epochs play a crucial role in the consensus algorithm and are used to determine the current validator set. They are also used as measurement in the staking mechanism.

Epoch can be read from the transaction context :

```
bash public fun current_epoch(ctx: &TxContext) { let epoch = ctx.epoch(); // ... }
```

It is also possible to get the unix timestamp of the epoch start:

```
bash public fun current_epoch_start(ctx: &TxContext) { let epoch_start = ctx.epoch_timestamp_ms();
// ... }
```

Normally, epochs are used in staking and system operations, however, in custom scenarios they can be used to emulate 24h periods. They are critical if an application relies on the staking logic or needs to know the current validator set.

For a more precise time measurement, Sui provides the Clock object. It is a system object that is updated during checkpoints by the system, which stores the current time in milliseconds since the Unix Epoch. The Clock object is defined in the sui::clock module and has a reserved address 0x6 .

Clock is a shared object, but a transaction attempting to access it mutably will fail. This limitation allows parallel access to the Clock object, which is important for maintaining performance.

```
bash // File: sui-framework/clock.move /// Singleton shared object that exposes time to Move calls.
This /// object is found at address 0x6, and can only be read (accessed /// via an immutable
reference) by entry functions. /// /// Entry Functions that attempt to accept `Clock` by mutable
/// reference or value will fail to verify, and honest validators /// will not sign or execute
transactions that use `Clock` as an /// input parameter, unless it is passed by immutable
reference. struct Clock has key { id: UID, /// The clock's timestamp, which is set automatically by
a /// system transaction every time consensus commits a /// schedule, or by
`sui::clock::increment_for_testing` during /// testing. timestamp_ms: u64, }
```

There is only one public function available in the Clock module - timestamp_ms . It returns the current time in milliseconds since the Unix Epoch.

```bash use sui::clock::Clock;
```

/// Clock needs to be passed as an immutable reference. public fun current_time(clock: &Clock) { let time = clock.timestamp_ms(); // ... } ```

Time

For a more precise time measurement, Sui provides the Clock object. It is a system object that is updated during checkpoints by the system, which stores the current time in milliseconds since the Unix Epoch. The Clock object is defined in the sui::clock module and has a reserved address 0x6 .

Clock is a shared object, but a transaction attempting to access it mutably will fail. This limitation allows parallel access to the Clock object, which is important for maintaining performance.

```
bash // File: sui-framework/clock.move /// Singleton shared object that exposes time to Move calls. This /// object is found at address 0x6, and can only be read (accessed /// via an immutable reference) by entry functions. /// /// Entry Functions that attempt to accept `Clock` by mutable /// reference or value will fail to verify, and honest validators /// will not sign or execute transactions that use `Clock` as an /// input parameter, unless it is passed by immutable reference. struct Clock has key { id: UID, /// The clock's timestamp, which is set automatically by a /// system transaction every time consensus commits a /// schedule, or by `sui::clock::increment_for_testing` during /// testing. timestamp_ms: u64, }
```

There is only one public function available in the Clock module - timestamp_ms . It returns the current time in milliseconds since the Unix Epoch.

```bash use sui::clock::Clock;

/// Clock needs to be passed as an immutable reference. public fun current_time(clock: &Clock) { let time = clock.timestamp_ms(); // ... } ```
```