Using Events

The Sui network stores countless objects on chain where Move code can perform actions using those objects. Tracking this activity is often desired, for example, to discover how many times a module mints an NFT or to tally the amount of SUI in transactions that a smart contract generates.

To support activity monitoring, Move provides a structure to emit events on the Sui network. You can then leverage a custom indexer to process checkpoint data that includes events that have been emitted. See the custom indexer topic in the Advanced section to learn how to stream checkpoints and filter events continuously.

If you don't want to run a custom indexer, you can poll the Sui network to query for emitted events instead. This approach typically includes a database to store the data retrieved from these calls. The Poll events section provides an example of using this method.

An event object in Sui consists of the following attributes:

To create an event in your Move modules, add the sui::event dependency.

With the dependency added, you can use the emit function to trigger an event whenever the action you want to monitor fires. For example, the following code is part of an example application that enables the locking of objects. The lock function handles the locking of objects and emits an event whenever the function is called.

The Sui RPC provides a queryEvents method to query on-chain packages and return available events. As an example, the following curl command queries the Deepbook package on Mainnet for a specific type of event:

A successful curl return

The TypeScript SDK provides a wrapper for the suix_queryEvents method: client.queryEvents .

TypeScript SDK queryEvents example

The example displays the following JSON representation of the event.

To filter the events returned from your queries, use the following data structures.

The Sui by Example repo on GitHub contains a code sample that demonstrates how to query events using the query_events function. The package that PACKAGE_ID_CONST points to exists on Mainnet, so you can test this code using Cargo. To do so, clone the sui-by-example repo locally and follow the Example 05 directions .

This content describes an alpha/beta feature or service. These early stage features and services are in active development, so details are likely to change.

You can use GraphQL to query events instead of JSON RPC. The following example queries are in the sui-graphql-rpc crate in the Sui repo.

Event connection

Filter events by sender

The TypeScript SDK provides functionality to interact with the Sui GraphQL service.

Firing events is not very useful in a vacuum. You also need the ability to respond to those events. There are two methods from which to choose when you need to monitor on-chain events:

Using a custom indexer provides a near-real time monitoring of events, so is most useful when your project requires immediate reaction to the firing of events. Polling the network is most useful when the events you're monitoring don't fire often or the need to act on those events are not immediate. The following section provides a polling example.

To monitor events, you need a database to store checkpoint data. The Trustless Swap example uses a Prisma database to store checkpoint data from the Sui network. The database is populated from polling the network to retrieve emitted events.

event-indexer.ts from Trustless Swap

Trustless Swap incorporates handlers to process each event type that triggers. For the locked event, the handler in locked-handler.ts fires and updates the Prisma database accordingly.

locked-handler.ts from Trustless Swap

Move event structure

An event object in Sui consists of the following attributes:

To create an event in your Move modules, add the sui::event dependency.

With the dependency added, you can use the emit function to trigger an event whenever the action you want to monitor fires. For example, the following code is part of an example application that enables the locking of objects. The lock function handles the locking of objects and emits an event whenever the function is called.

The Sui RPC provides a queryEvents method to query on-chain packages and return available events. As an example, the following curl command queries the Deepbook package on Mainnet for a specific type of event:

A successful curl return

The TypeScript SDK provides a wrapper for the suix_queryEvents method: client.queryEvents .

TypeScript SDK queryEvents example

The example displays the following JSON representation of the event.

To filter the events returned from your queries, use the following data structures.

The Sui by Example repo on GitHub contains a code sample that demonstrates how to query events using the query_events function. The package that PACKAGE_ID_CONST points to exists on Mainnet, so you can test this code using Cargo. To do so, clone the sui-by-example repo locally and follow the Example 05 directions .

This content describes an alpha/beta feature or service. These early stage features and services are in active development, so details are likely to change.

You can use GraphQL to query events instead of JSON RPC. The following example queries are in the sui-graphql-rpc crate in the Sui repo.

Event connection

Filter events by sender

The TypeScript SDK provides functionality to interact with the Sui GraphQL service.

Firing events is not very useful in a vacuum. You also need the ability to respond to those events. There are two methods from which to choose when you need to monitor on-chain events:

Using a custom indexer provides a near-real time monitoring of events, so is most useful when your project requires immediate reaction to the firing of events. Polling the network is most useful when the events you're monitoring don't fire often or the need to act on those events are not immediate. The following section provides a polling example.

To monitor events, you need a database to store checkpoint data. The Trustless Swap example uses a Prisma database to store checkpoint data from the Sui network. The database is populated from polling the network to retrieve emitted events.

event-indexer.ts from Trustless Swap

Trustless Swap incorporates handlers to process each event type that triggers. For the locked event, the handler in locked-handler.ts fires and updates the Prisma database accordingly.

locked-handler.ts from Trustless Swap

Emit events in Move

To create an event in your Move modules, add the sui::event dependency.

With the dependency added, you can use the emit function to trigger an event whenever the action you want to monitor fires. For example, the following code is part of an example application that enables the locking of objects. The lock function handles the

locking of objects and emits an event whenever the function is called.

The Sui RPC provides a queryEvents method to query on-chain packages and return available events. As an example, the following curl command queries the Deepbook package on Mainnet for a specific type of event:

A successful curl return

The TypeScript SDK provides a wrapper for the suix_queryEvents method: client.queryEvents .

TypeScript SDK queryEvents example

The example displays the following JSON representation of the event.

To filter the events returned from your queries, use the following data structures.

The Sui by Example repo on GitHub contains a code sample that demonstrates how to query events using the query_events function. The package that PACKAGE_ID_CONST points to exists on Mainnet, so you can test this code using Cargo. To do so, clone the sui-by-example repo locally and follow the Example 05 directions .

This content describes an alpha/beta feature or service. These early stage features and services are in active development, so details are likely to change.

You can use GraphQL to query events instead of JSON RPC. The following example queries are in the sui-graphql-rpc crate in the Sui repo.

Event connection

Filter events by sender

The TypeScript SDK provides functionality to interact with the Sui GraphQL service.

Firing events is not very useful in a vacuum. You also need the ability to respond to those events. There are two methods from which to choose when you need to monitor on-chain events:

Using a custom indexer provides a near-real time monitoring of events, so is most useful when your project requires immediate reaction to the firing of events. Polling the network is most useful when the events you're monitoring don't fire often or the need to act on those events are not immediate. The following section provides a polling example.

To monitor events, you need a database to store checkpoint data. The Trustless Swap example uses a Prisma database to store checkpoint data from the Sui network. The database is populated from polling the network to retrieve emitted events.

event-indexer.ts from Trustless Swap

Trustless Swap incorporates handlers to process each event type that triggers. For the locked event, the handler in locked-handler.ts fires and updates the Prisma database accordingly.

locked-handler.ts from Trustless Swap

Monitoring events

Firing events is not very useful in a vacuum. You also need the ability to respond to those events. There are two methods from which to choose when you need to monitor on-chain events:

Using a custom indexer provides a near-real time monitoring of events, so is most useful when your project requires immediate reaction to the firing of events. Polling the network is most useful when the events you're monitoring don't fire often or the need to act on those events are not immediate. The following section provides a polling example.

To monitor events, you need a database to store checkpoint data. The Trustless Swap example uses a Prisma database to store checkpoint data from the Sui network. The database is populated from polling the network to retrieve emitted events.

event-indexer.ts from Trustless Swap

Trustless Swap incorporates handlers to process each event type that triggers. For the locked event, the handler in locked-handler.ts fires and updates the Prisma database accordingly.

locked-handler.ts from Trustless Swap

Related links