

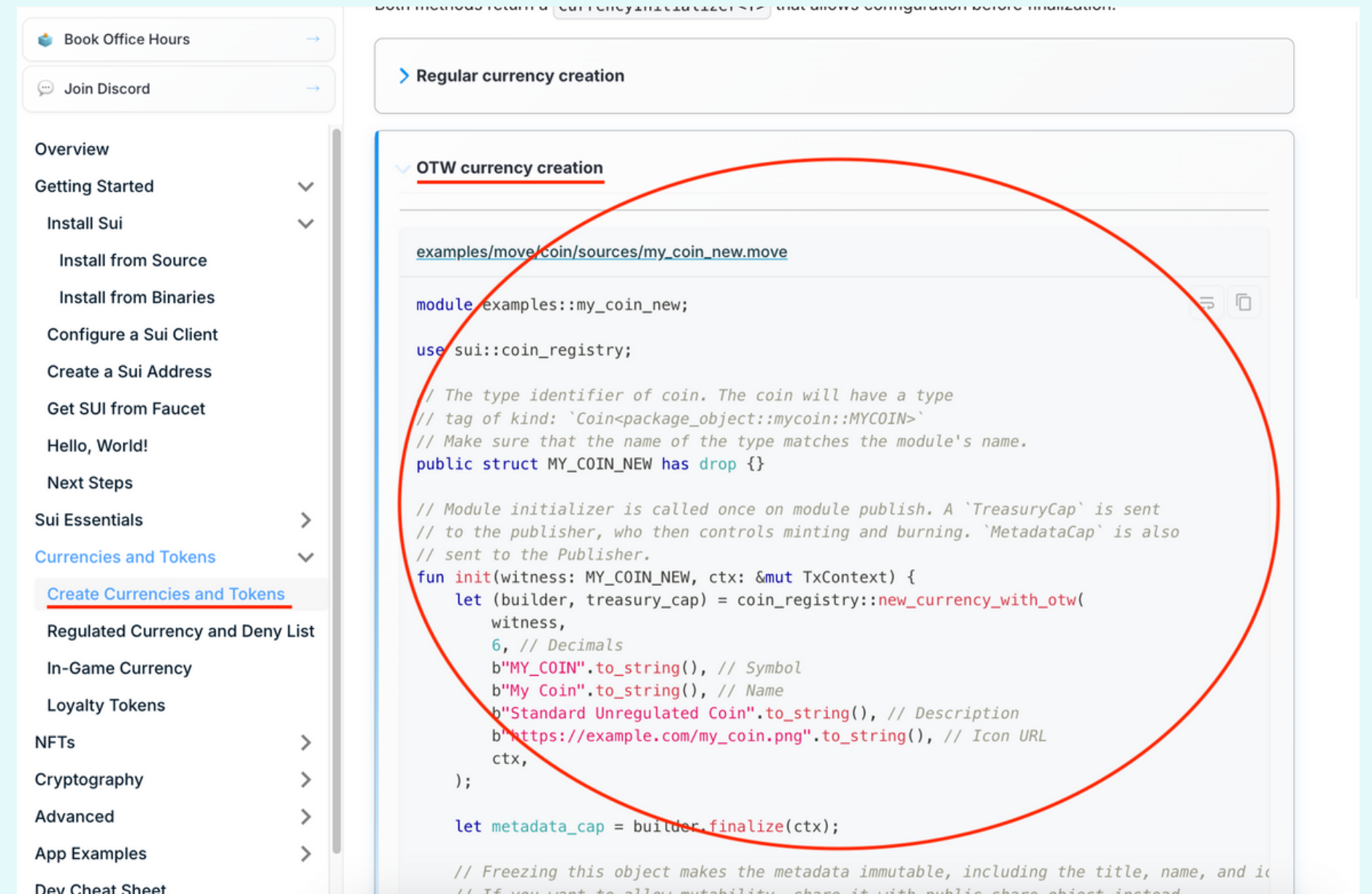
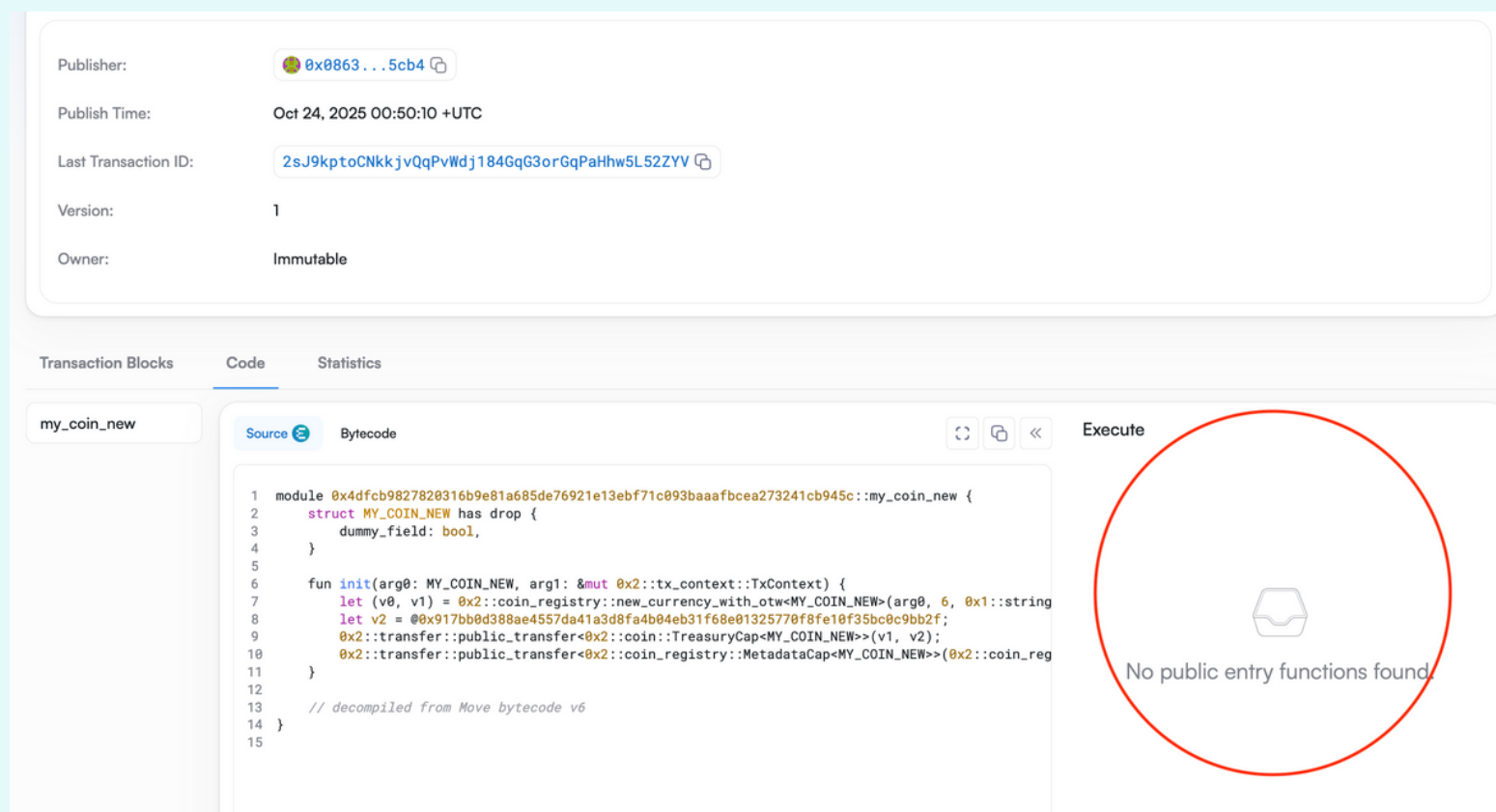
Build on Sui
Weekend Move Workshop
in Tokyo // Day 2

Powered by



開発25 public, entry関数

mintを実行しようとしたのですが、**public entry関数がない**と言われました。



<https://docs.sui.io/guides/developer/currency>

開発26 entry関数の作成

以下のようにmint関数を作ります。

<MY_COIN_NEW>で型を設定しています。

```
Home my_coin_new.move X Object Explorer
1  module empty::my_coin_new;
2
3  use sui::coin;
4  use sui::coin_registry;
5
6  public struct MY_COIN_NEW has drop {}
7
8  > fun init(witness: MY_COIN_NEW, ctx: &mut TxContext) { ...
22 }
23
24 ∨ entry fun mint(
25     cap: &mut coin::TreasuryCap<MY_COIN_NEW>,
26     amount: u64,
27     recipient: address,
28     ctx: &mut TxContext
29 ∨ ) {
30     let new_coin = coin::mint<MY_COIN_NEW>(cap, amount, ctx);
31     transfer::public_transfer(new_coin, recipient);
32 }
```

開発27 SuiVisionでの実行

SuiVisionで実行ができることを確認しましょう。
あれ、できない？

Publisher:

0x0863...5cb4

Publish Time:

Oct 24, 2025 02:22:05 +UTC

Last Transaction ID:

FfmB8pvnpKDAHGMfu3PSHarMTWALjULdzV1HGbfMgtLe

Version:

1

Owner:

Immutable

Transaction Blocks

Code

Statistics

my_coin_new

Source

Bytecode

```
1 module 0xbde479c784c71eb37fed2005cfbdb9f555adff2469ac9791524a94a0ae0e7865::my_coin_new {
2   struct MY_COIN_NEW has drop {
3     dummy_field: bool,
4   }
5
6   entry fun mint(arg0: &mut 0x2::coin::TreasuryCap<MY_COIN_NEW>, arg1: u64, arg2: address, arg3:
7     0x2::transfer::public_transfer<0x2::coin::Coin<MY_COIN_NEW>>(0x2::coin::mint<MY_COIN_NEW>{
8   }
9
10  fun init(arg0: MY_COIN_NEW, arg1: &mut 0x2::tx_context::TxContext) {
11    let (v0, v1) = 0x2::coin_registry::new_currency_with_otw<MY_COIN_NEW>(arg0, 6, 0x1::string
12      0x2::transfer::public_transfer<0x2::coin::TreasuryCap<MY_COIN_NEW>>(v1, 0x2::tx_context::s
13      0x2::transfer::public_transfer<0x2::coin_registry::MetadataCap<MY_COIN_NEW>>(0x2::coin_reg
14    )
15  }
16  // decompiled from Move bytecode v6
17 }
18 }
```

Execute

mint

Arg0

0x6af242069e8718fa4fbd54f78527d64c4937193b:

Arg1

1111

Arg2

0x884344b72b6d2d20684b91ce84beaf8b8cbce8:

Execute

SuiVisionBlockchainValidators

Ox0863...5cb4

Search by Account, Package, Object, Transaction, SuiNS

Object

0x96dd...5dfb

Details

Owner:

0x8843...d33b

Publisher:

0x0863...5cb4

Type:

0x2::coin::Coin<0xbde4...7865::my_coin_new::MY_COIN_NEW>

Version:

623364027

Transaction Block Digest:

G1wKwP...ePp8

Fields

```
{
  balance: "1111"
  id: {
    id: "0x96dd0504599d940dab5d783bf6121ffb60dc9057685732d39becaece5fdf5dfb"
  }
}
```

Dynamic Fields

CHILD OBJECT ID	VERSION	OBJECT TYPE	LAST TX ID
-----------------	---------	-------------	------------

開発27 SuiVisionでの実行

実行者が異なっていたのですね。

```
14 fun init(witness: MY_COIN_NEW, ctx: &mut TxContext) {
15     let (builder: CurrencyInitializer, treasury_cap: TreasuryCap) = coin_registry::new_currency_w
16     let witness =
17         symbol: std::string::String
18         symbol: b"MY_COIN".to_string(), // Symbol
19         name: b"My Coin".to_string(), // Name
20         description: b"Standard Unregulated Coin".to_string(), // Description
21         icon_url: b"https://example.com/my_coin.png".to_string(), // Icon URL
22         ctx: ctx,
23     };
24
25     let metadata_cap: MetadataCap = builder.finalize(ctx: ctx);
26
27     // Freezing this object makes the metadata immutable, including the title, name, and icon ima
28     // If you want to allow mutability, share it with public_share_object instead.
29
30     let my_account : address : address = @0x0863a92345a2f9f9fc88e659cbcc983b2512425a2f2ffe02edbb5d
31     // transfer::public_transfer(treasury_cap, ctx.sender());
32     // transfer::public_transfer(metadata_cap, ctx.sender());
33     transfer::public_transfer(obj: obj: treasury_cap, recipient: recipient: my_account);
34     transfer::public_transfer(obj: obj: metadata_cap, recipient: recipient: my_account);
35 }
```

https://github.com/ytakahashi2020/my_coin/blob/main/sources/my_coin.move

セキュリティの考慮事項

ガス最適化

高コストなトランザクションによるサービス拒否を避けるため、コストを最小化します。

リエントランシー対策

リソースモデルによりリエントランシー攻撃を効率的に阻止します。

形式検証

重要なコントラクトロジックの正しさを数学的に証明します。

型からの派生 ⑥ガス最適化

Ethereumの場合、計算コストとストレージコストが合算してガスPriceにかけていますが、**Solana・Suiは単価を分けて**います。

Solanaの場合は、事前にレンタル代として確保するので、書き方を変えています。

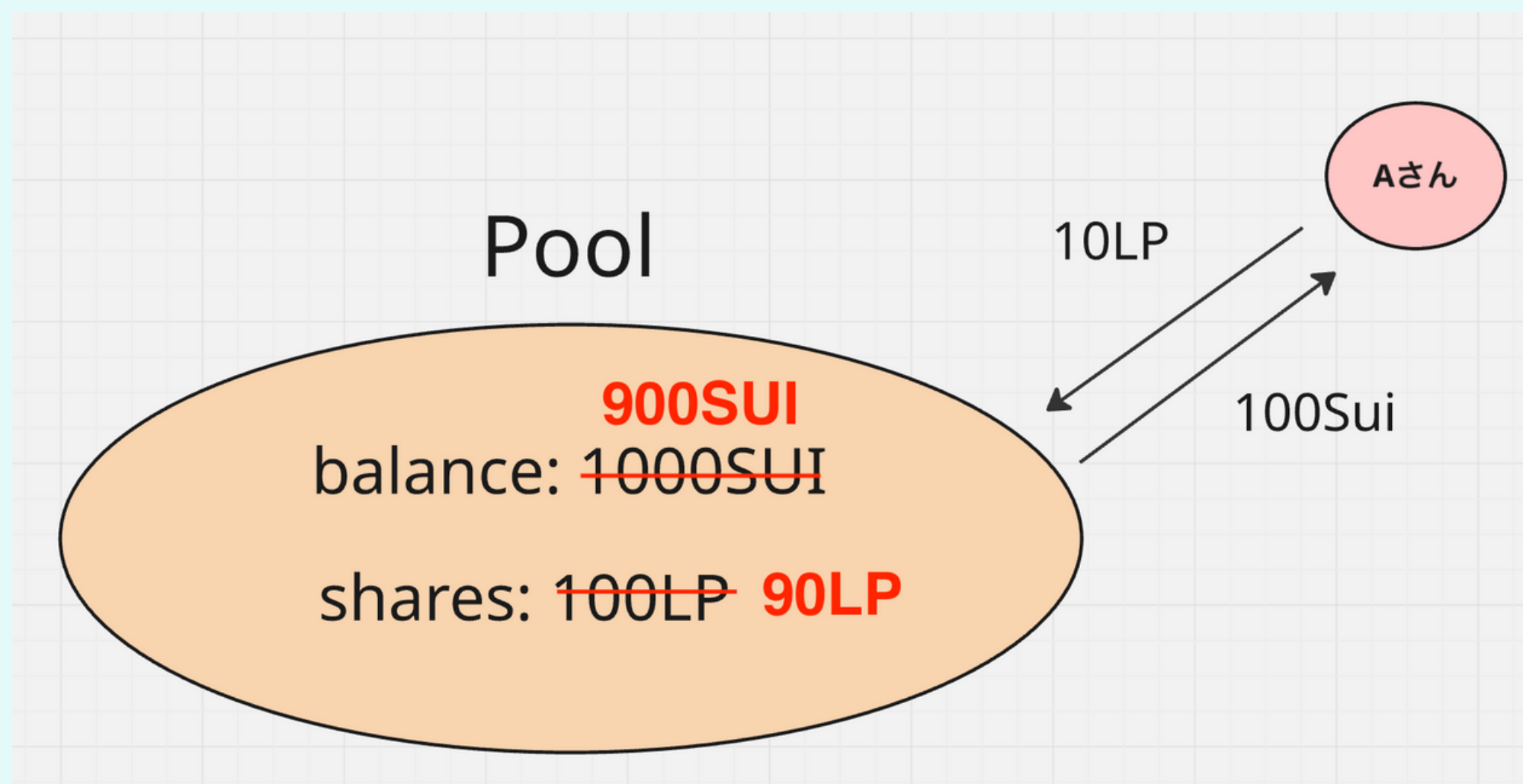
Suiでは**オブジェクトで所有者が決まっている**ので、**ストレージの計算が明瞭**です。

Ethereum		
ガス使用量 × ガスPrice		
Solana		
署名数 × 基本手数料 + (CUPrice × 設定CU)	+	(data_length × rent_exemption_rate)
Sui		
computation_units × reference_gas_price	+	storage_units × storage_price

型からの派生 ⑦形式的検証

プールからの引き出し（withdraw）について考えてみます。

例えば、所有している10LPを使用することで、その割合分である100SUIが取得できるとします。

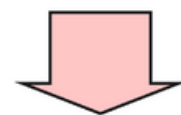


型からの派生 ⑦形式的検証

今回は、引き出し後のLPの価値が引き出し前の価値より下がらないことを証明します。

割り算は使いたくないので、下のような掛け算の内容を証明します。

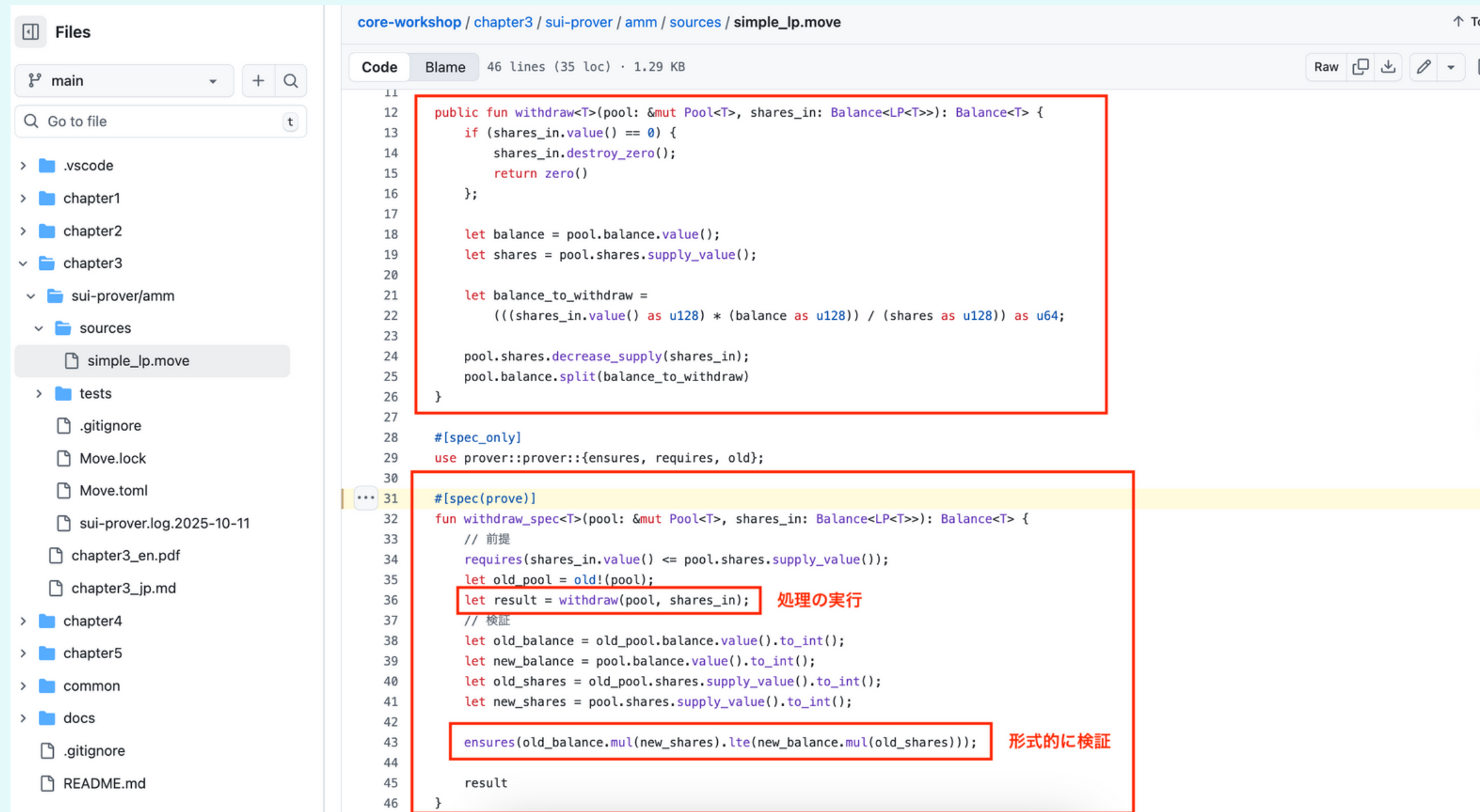
$$\frac{\text{引き出し前の総額 (1000SUI)}}{\text{引き出し前のシェア (100LP)}} \leq \frac{\text{引き出し後の総額 (900SUI)}}{\text{引き出し後のシェア (90LP)}}$$



$$\begin{aligned} & \text{引き出し前の総額 (1000SUI)} \times \text{引き出し後のシェア (90LP)} \\ & \leq \text{引き出し後の総額 (900SUI)} \times \text{引き出し前のシェア (100LP)} \end{aligned}$$

型からの派生 ⑦形式的検証

以下のように、実行後に数学的に検証します。



The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with folders like 'chapter3' and 'sources', and files like 'simple_lp.move'. The code editor shows the implementation of a withdrawal function and its formal verification. The implementation is highlighted with a red box, and the verification is highlighted with a yellow box. The verification includes a pre-condition, the function call, and a post-condition.

```
11
12 public fun withdraw<T>(pool: &mut Pool<T>, shares_in: Balance<LP<T>>): Balance<T> {
13     if (shares_in.value() == 0) {
14         shares_in.destroy_zero();
15         return zero()
16     };
17
18     let balance = pool.balance.value();
19     let shares = pool.shares.supply_value();
20
21     let balance_to_withdraw =
22         (((shares_in.value() as u128) * (balance as u128)) / (shares as u128)) as u64;
23
24     pool.shares.decrease_supply(shares_in);
25     pool.balance.split(balance_to_withdraw)
26 }
27
28 #[spec_only]
29 use prover::prover::{ensures, requires, old};
30
31 ...
32 #[spec(prove)]
33 fun withdraw_spec<T>(pool: &mut Pool<T>, shares_in: Balance<LP<T>>): Balance<T> {
34     // 前提
35     requires(shares_in.value() <= pool.shares.supply_value());
36     let old_pool = old!(pool);
37     let result = withdraw(pool, shares_in);
38     // 検証
39     let old_balance = old_pool.balance.value().to_int();
40     let new_balance = pool.balance.value().to_int();
41     let old_shares = old_pool.shares.supply_value().to_int();
42     let new_shares = pool.shares.supply_value().to_int();
43     ensures(old_balance.mul(new_shares).lte(new_balance.mul(old_shares)));
44
45     result
46 }
```

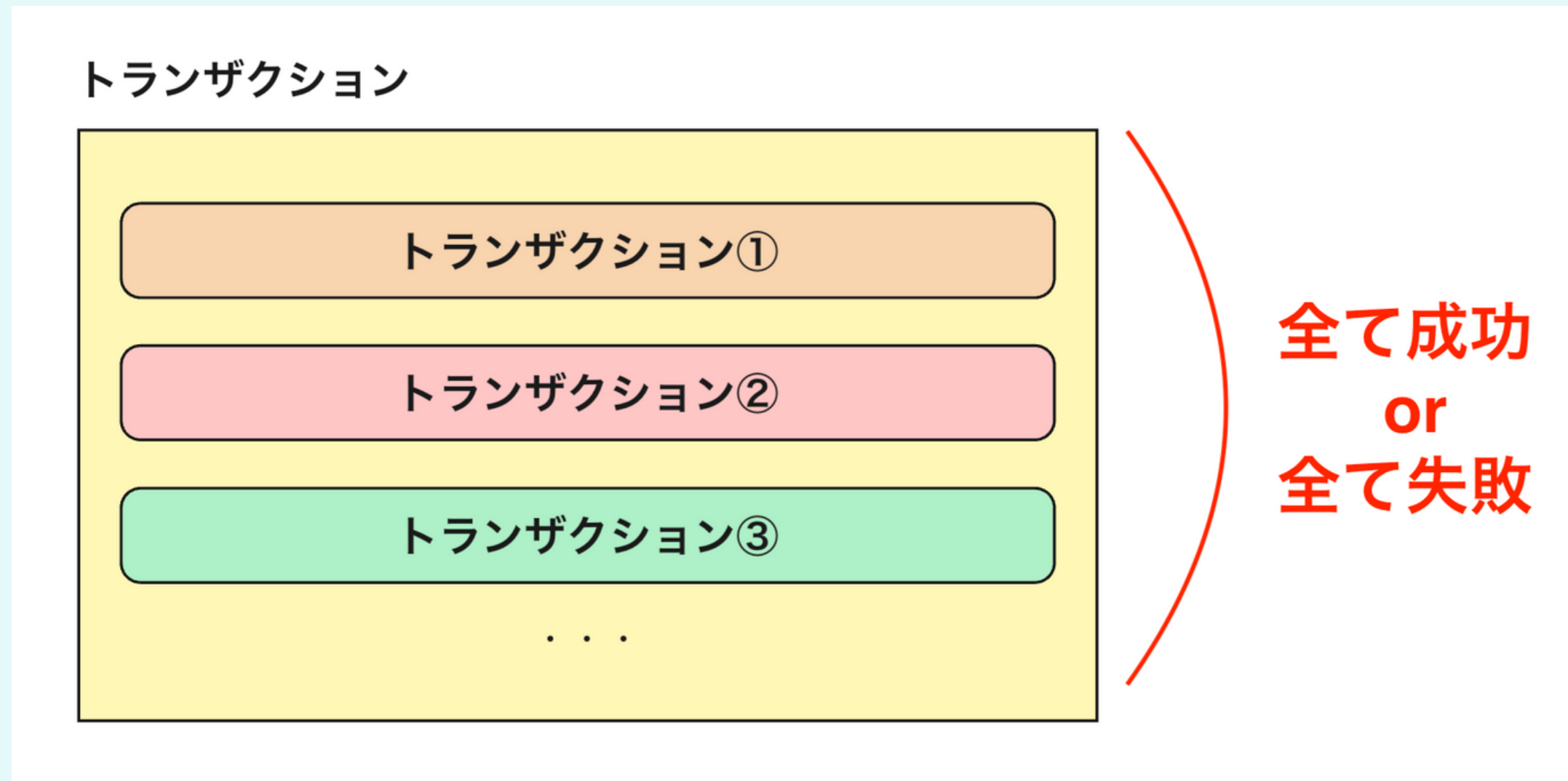
処理の実行

形式的に検証

https://github.com/SuiJapan/core-workshop/blob/main/chapter3/sui-prover/amm/sources/simple_lp.move#L31

PTBとは

PTB (programmable transaction block) はプログラム可能なトランザクションのブロックです。
まとめることで**原子性 (全て成功か失敗)** を確保できます。



PTBとは

また、大きな特徴として、**前の結果をそのまま次に渡せ**ます。

トランザクション

1 コインの分割

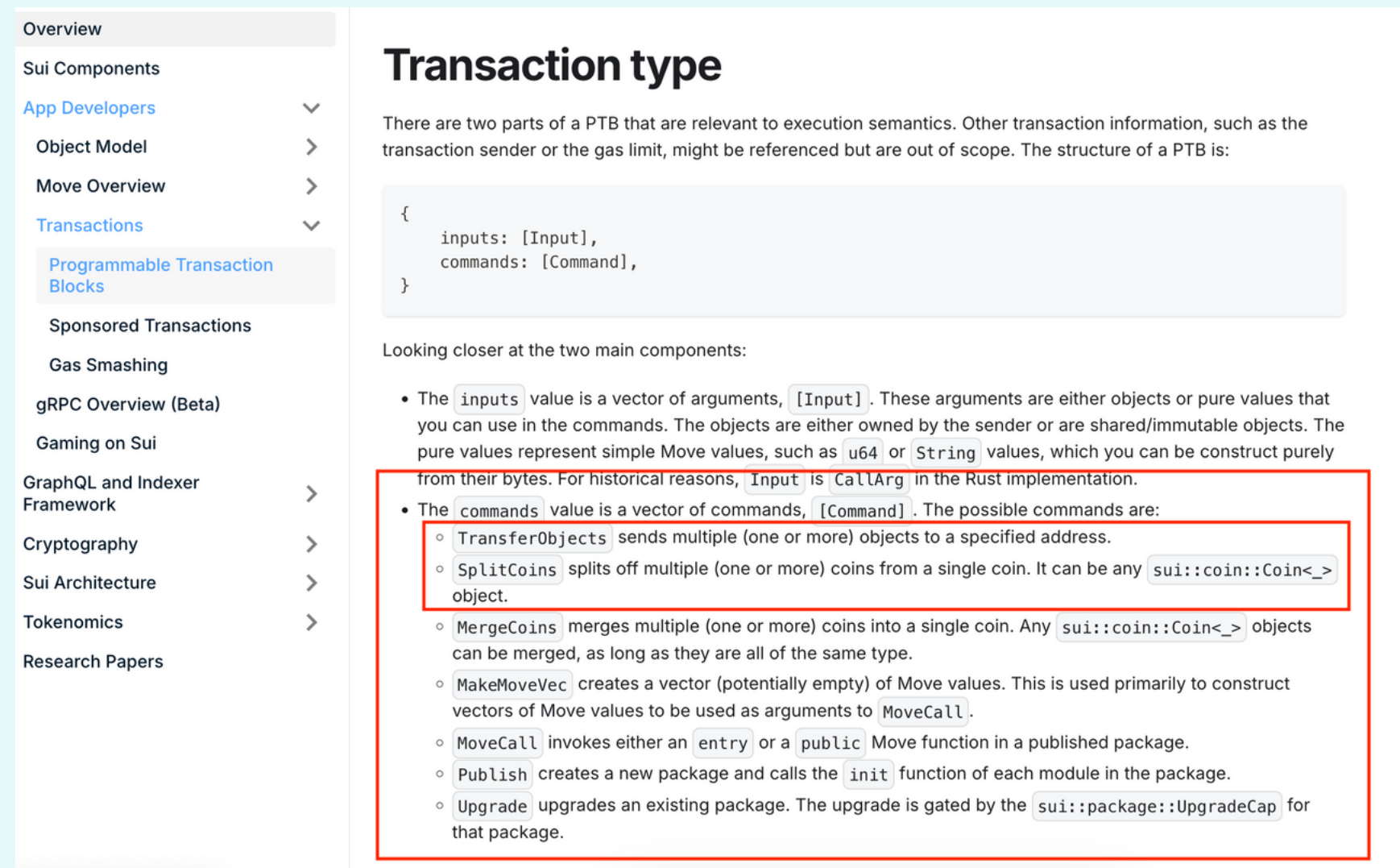


2 分割したコインの送付

PTBとは

こちらが使用できるコマンドです。

今回は「**TransferObjects**」と「**SplitCoins**」を使います。



The screenshot shows the Sui documentation page for Transaction type. The left sidebar contains a navigation menu with items like Overview, Sui Components, App Developers, Object Model, Move Overview, Transactions, Programmable Transaction Blocks, Sponsored Transactions, Gas Smashing, gRPC Overview (Beta), Gaming on Sui, GraphQL and Indexer Framework, Cryptography, Sui Architecture, Tokenomics, and Research Papers. The main content area is titled "Transaction type" and explains that there are two parts of a PTB relevant to execution semantics. It shows the structure of a PTB as a JSON object with inputs and commands. Below this, it lists the two main components: inputs and commands. The commands section is highlighted with a red box and lists several commands: TransferObjects, SplitCoins, MergeCoins, MakeMoveVec, MoveCall, Publish, and Upgrade. The TransferObjects and SplitCoins commands are further detailed with their functions and parameters.

Transaction type

There are two parts of a PTB that are relevant to execution semantics. Other transaction information, such as the transaction sender or the gas limit, might be referenced but are out of scope. The structure of a PTB is:

```
{
  inputs: [Input],
  commands: [Command],
}
```

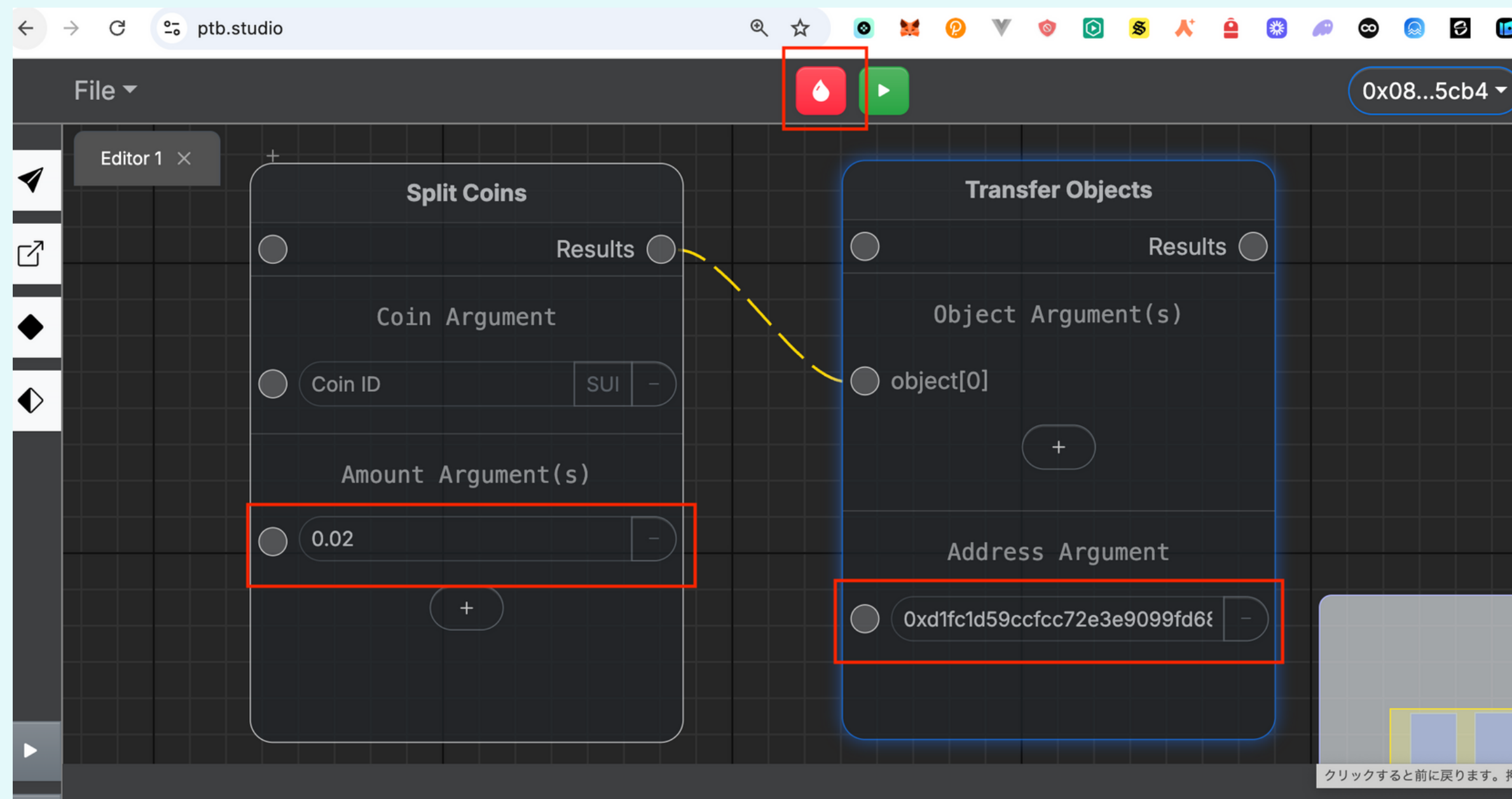
Looking closer at the two main components:

- The `inputs` value is a vector of arguments, `[Input]`. These arguments are either objects or pure values that you can use in the commands. The objects are either owned by the sender or are shared/immutable objects. The pure values represent simple Move values, such as `u64` or `String` values, which you can construct purely from their bytes. For historical reasons, `Input` is `CallArg` in the Rust implementation.
- The `commands` value is a vector of commands, `[Command]`. The possible commands are:
 - `TransferObjects` sends multiple (one or more) objects to a specified address.
 - `SplitCoins` splits off multiple (one or more) coins from a single coin. It can be any `sui::coin::Coin<_>` object.
 - `MergeCoins` merges multiple (one or more) coins into a single coin. Any `sui::coin::Coin<_>` objects can be merged, as long as they are all of the same type.
 - `MakeMoveVec` creates a vector (potentially empty) of Move values. This is used primarily to construct vectors of Move values to be used as arguments to `MoveCall`.
 - `MoveCall` invokes either an `entry` or a `public` Move function in a published package.
 - `Publish` creates a new package and calls the `init` function of each module in the package.
 - `Upgrade` upgrades an existing package. The upgrade is gated by the `sui::package::UpgradeCap` for that package.

<https://docs.sui.io/concepts/transactions/prog-txn-blocks>

PTBとは

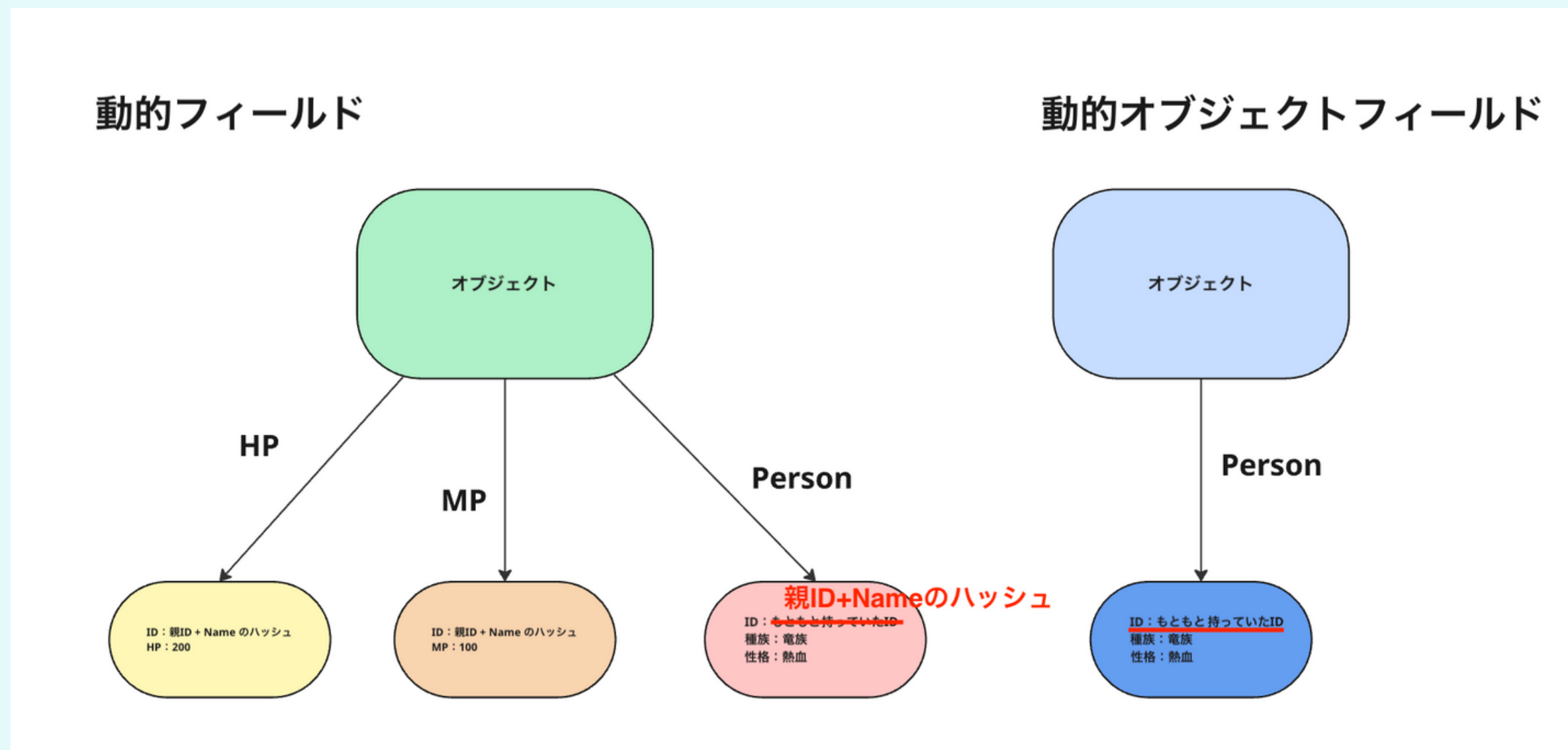
ptb.studioで体験してみましょう。



<https://ptb.studio/>

動的（オブジェクト）フィールド

Suiではオブジェクトに動的にフィールドの設定ができます。
2種類がありますが、どちらもオブジェクトとして付属させます。



https://github.com/ytakahashi2020/dynamic_field/blob/main/my/sources/dynamic_field_demo.move

動的（オブジェクト） フィールド

例えば、下のオブジェクトは「HP」,「MP」というフィールドを持ち、それぞれがオブジェクトです。

Object
0xd66d...d385

Search by Account, Package, Object, Transaction, SuiNS

Details

Owner: 0x0863...5cb4
Publisher: 0x0863...5cb4
Type: 0xe051...fb8b::simple...demo::Parent
Version: 619109403
Transaction Block Digest: 46wts7...QGp7

Fields

```
{  
  id: {  
    id: "0xd66dbd412bbefef054549ca2bc650a23782ec0ed552948de515f983c2aacd385"  
  }  
}
```

Dynamic Fields

CHILD OBJECT ID	VERSION	OBJECT TYPE	LAST TX ID	
0x214f...9e1f	619109403	u64	46wts7...QGp7	HPフィールド
0x8587...9828	619109402	u64	DetrjF...axQP	MPフィールド

Object
0x214f...9e1f

Search by Account, Package, Object, Transaction, SuiNS

Details

Owner: 0xd66d...d385
Publisher: 0x0863...5cb4
Type: 0x2::dynamic_field::Field<vector<u8>, u64>
Version: 619109403
Transaction Block Digest: 46wts7...QGp7

Fields

```
{  
  id: {  
    id: "0x214fba91545b10bdd1a860dfb386475c475af57b99a699d6ee151328b8cb9e1f"  
  }  
  name: {  
    0: 104 H  
    1: 112 P  
  }  
  value: "200"  
}
```

Dynamic Fields

CHILD OBJECT ID	VERSION	OBJECT TYPE	LAST TX ID
-----------------	---------	-------------	------------

<https://testnet.suivision.xyz/object/0xa8fc841714fef0a9dcabdca7ba7c98a678359180a6984bfcc4a552913d2d93e0f>

ガスレストランザクション

ガススポンサーの役割

- ✓ ユーザー、ガスステーション、スポンサー間で手数料をシームレスに管理します。

サービス例

- ✓ ShinamiのGas Stationを利用することで、ガス代のスポンサーが容易になります。

<https://blog.sui.io/shinami-gas-station-tutorial/>

ユースケース

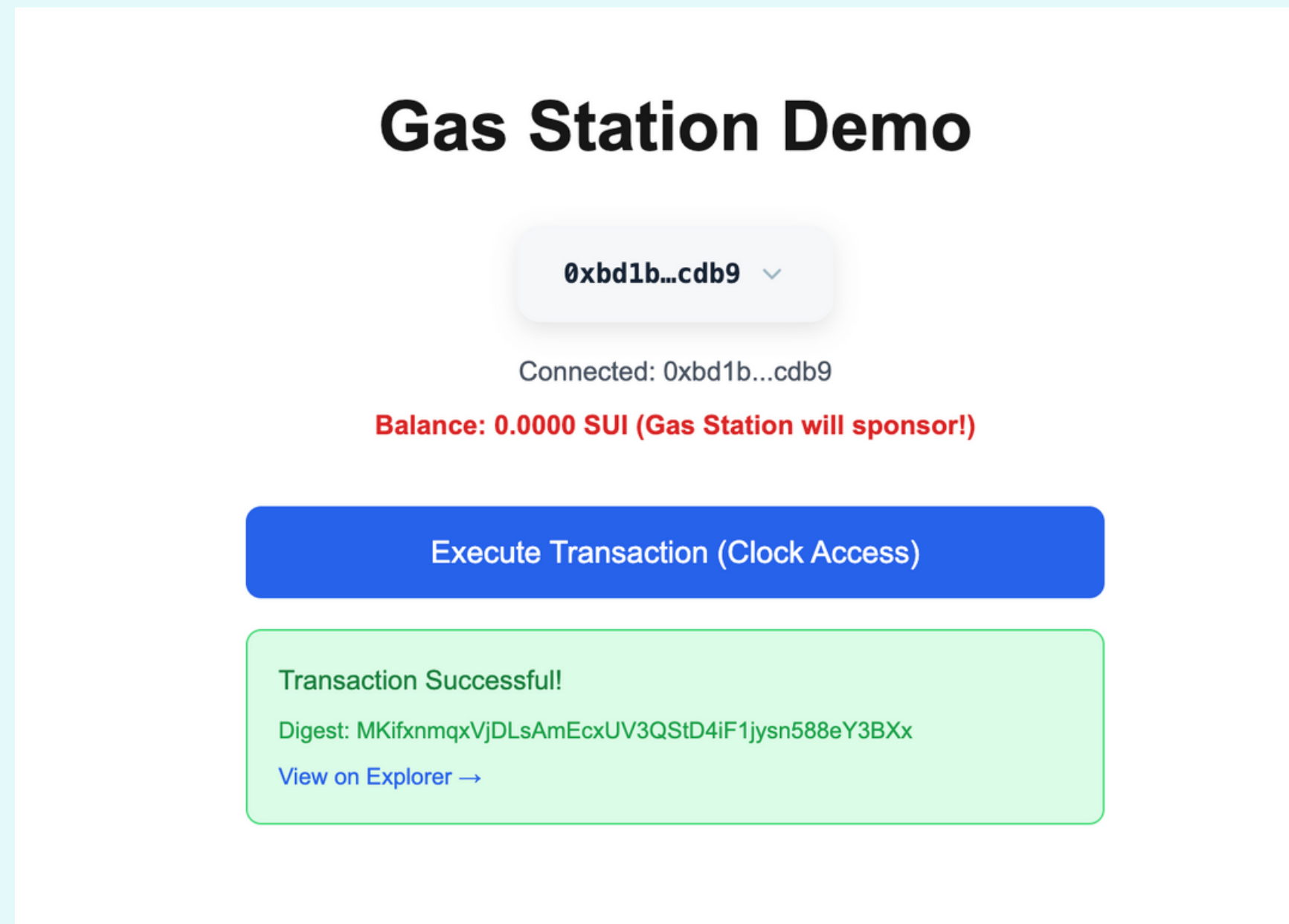
- ✓ ゲームdAppが初期トランザクションのガス代を負担し、ユーザー獲得を促進します。

演習

- ✓ スポンサーを設定します。
- ✓ ガスレストランザクションを送信します。
- ✓ レシートとガス支払者をオンチェーンで確認します。

1 ガスレストラランザクション

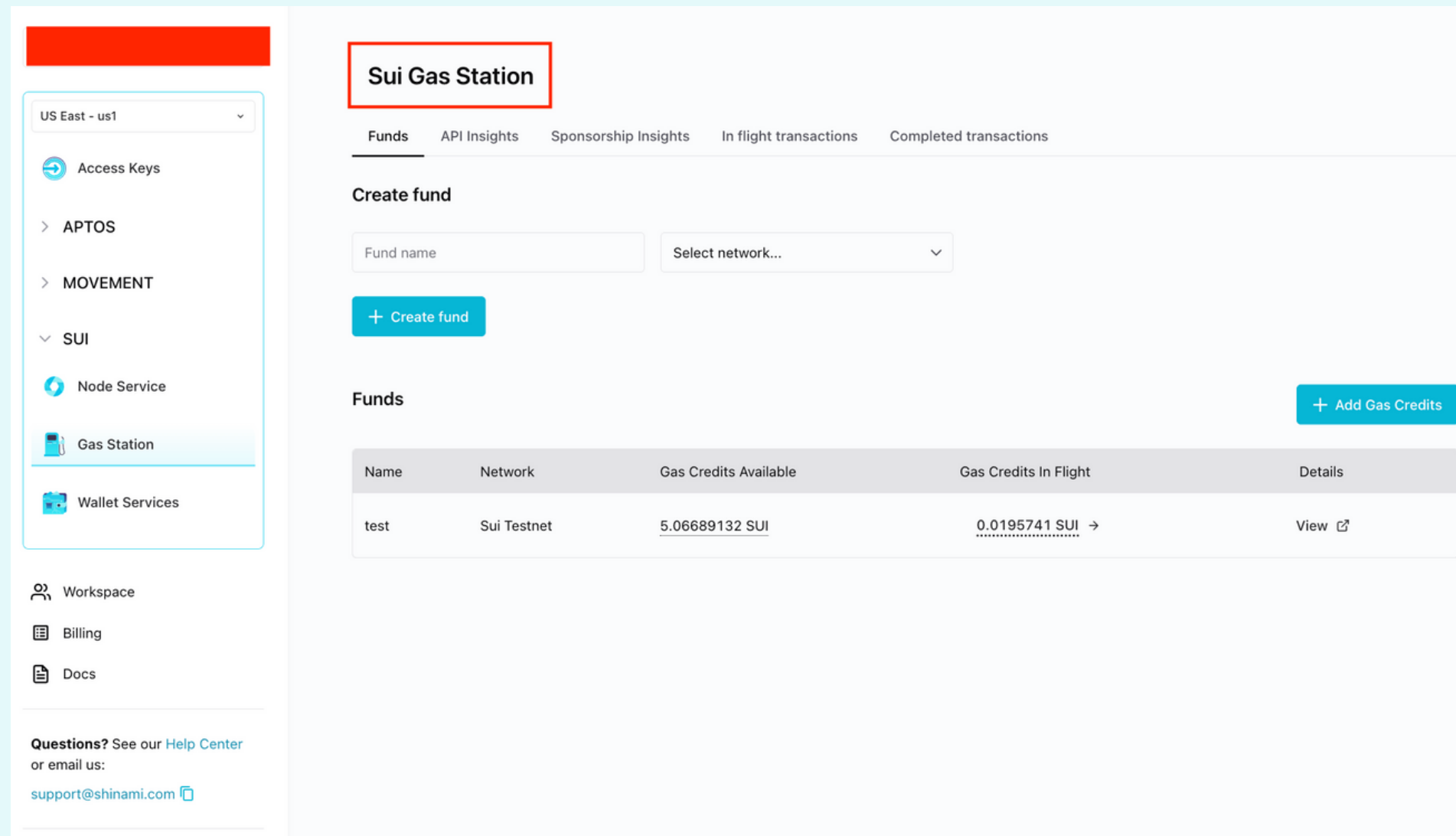
管理者がガス代を負担することで、利用者はガス代なしで実行ができます。



<https://sui-gas-station-demo.vercel.app/>

1 ガスレストラランザクション

Shinamiを使うことで簡単に実装できます。
Gas Stationにガスを入れます。



The screenshot displays the Shinami application interface. On the left is a sidebar with a navigation menu including 'Access Keys', 'APTOS', 'MOVEMENT', 'SUI' (expanded), 'Node Service', 'Gas Station' (highlighted), and 'Wallet Services'. Below this are links for 'Workspace', 'Billing', and 'Docs', along with a 'Help Center' link and email address 'support@shinami.com'. The main content area is titled 'Sui Gas Station' (highlighted with a red box) and features tabs for 'Funds', 'API Insights', 'Sponsorship Insights', 'In flight transactions', and 'Completed transactions'. The 'Funds' tab is active, showing a 'Create fund' section with input fields for 'Fund name' and 'Select network...', and a '+ Create fund' button. Below this is a table of funds with columns: Name, Network, Gas Credits Available, Gas Credits In Flight, and Details. A '+ Add Gas Credits' button is located at the top right of the table.

Name	Network	Gas Credits Available	Gas Credits In Flight	Details
test	Sui Testnet	5.06689132 SUI	0.0195741 SUI →	View ↗

<https://app.shinami.com/>

1 ガスレストランザクション

APIキーを取得することで簡単に実装できます。

Access keys [+ Create key](#) [Enable](#) [Disable](#) [Delete](#)

	Name / Key	Status	Chain	Network	Produ
☐	test1 ✎ us1_sui_.....fb11d 📄 (FOR BACKEND USE)	Enabled	Sui	Sui Testnet	N G

Node Service Gas Station Access Control

QPS limit: Allotment in use: **2 / 50** WebSocket subscriptions: Allotment in use: **1 / 40** [Save](#)

JSON-RPC API [API Doc & sample requests](#)

Sui Node URL: <https://api.us1.shinami.com/sui/node/v1>

Curl [Curl URL Auth](#) [Shinami Typescript SDK](#)

```
curl https://api.us1.shinami.com/sui/node/v1 \
-X POST \
-H 'X-API-Key: [REDACTED]' \
-H 'Content-Type: application/json' \
-d '{ "jsonrpc": "2.0", "method": "suix_getReferenceGasPrice", "params": [], "id": 1 }'
```

1 ガスレストランザクション

そもそもの**スポンサードトランザクション**は**5つのステップ**からなります。

Create a user-initiated sponsored transaction

A user-initiated sponsored transaction involves the following steps:

1. A user initializes a `GasLessTransactionData` transaction. ①ユーザーが「ガスレストランザクションデータ」を作る
2. The user sends `GasLessTransactionData` to the sponsor. ②ユーザーが①をスポンサーに送る
3. The sponsor validates the transaction, constructs `TransactionData` with gas fees, and then signs `TransactionData`. ③スポンサーはガスをつけて「トランザクションデータ」を作り、署名する
4. The sponsor sends the signed `TransactionData` and the sponsor `Signature` back to the user. ④スポンサーは③とその署名をユーザーに送る
5. The user verifies and then signs `TransactionData` and sends the dual-signed transaction to Sui network through a full node or the sponsor. ⑤ユーザーは③を署名し、スポンサーの署名とともにSuiネットワークに送る

<https://docs.sui.io/concepts/transactions/sponsored-transactions>

1 ガスレストランザクション

コードを読み解いてみましょう。

```
1 /
2
3 8 ✓ async function main() {
4   9   const node = createSuiClient(ACCESS_KEY); // Node Service
5   10  const gas = new GasStationClient(ACCESS_KEY); // Gas Station
6   11  const keypair = Ed25519Keypair.fromSecretKey(SENDER_SECRET);
7   12
8   13  // 1) ガス無しTxKindを作る (時計アクセス)
9   14  const tx = new Transaction();
10  15  tx.moveCall({
11  16    target:
12  17    "0xfa0e78030bd16672174c2d6cc4cd5d1d1423d03c28a74909b2a148eda8bcca16::clock::access",
13  18    arguments: [tx.object("0x6")],
14  19  });
15  20  const txKindBytes = await tx.build({
16  21    client: node,
17  22    onlyTransactionKind: true,
18  23  });
19  24  const txKindB64 = Buffer.from(txKindBytes).toString("base64");
20  25
21  26  // 2) スポンサー取得 (Auto-budget)
22  27  const sponsorship = await gas.sponsorTransaction({
23  28    txKind: txKindB64,
24  29    sender: keypair.toSuiAddress(),
25  30  });
26  31
27  32  // 3) 送信者署名
28  33  const signedBySender = await Transaction.from(sponsorship.txBytes).sign({
29  34    signer: keypair,
30  35  });
31  36
32  37  // 4) 送信 (スポンサー署名 + 送信者署名)
33  38  const exec = await node.executeTransactionBlock({
34  39    transactionBlock: sponsorship.txBytes,
35  40    signature: [signedBySender.signature, sponsorship.signature],
36  41  });
37  42
38  43  console.log("txDigest:", exec.digest);
39  44  }
40  45
```

https://github.com/SuiJapan/core-workshop/blob/main/chapter5/shinami/gas_station_min.ts

1 ガスレストラランザクション

- `npm init -y`でpackage.json作成
- `touch index.js`でファイル作成
- APIキー、プライベートキーを設定
- `npm i @shinami/clients @mysten/sui/keypairs`

1 ガスレストランザクション

①ガスレストランザクションデータの作成
ポイントは**onlyTransactionKind: true**です。

```
// 1) ガス無しTxKindを作る (時計アクセス)
const tx = new Transaction();
tx.moveCall({
  target:
    "0xfa0e78030bd16672174c2d6cc4cd5d1d1423d03c28a74909b2a148eda8bcca16::clock::access",
  arguments: [tx.object("0x6")],
});
const txKindBytes = await tx.build({
  client: node,
  onlyTransactionKind: true,
});
const txKindB64 = Buffer.from(txKindBytes).toString("base64");

// 2) スポンサー取得 (Auto-budget)
const sponsorship = await gas.sponsorTransaction({
  txKind: txKindB64,
  sender: keypair.toSuiAddress(),
});

// 3) 送信者署名
const signedBySender = await Transaction.from(sponsorship.txBytes).sign({
  signer: keypair,
});

// 4) 送信 (スポンサー署名 + 送信者署名)
const exec = await node.executeTransactionBlock({
  transactionBlock: sponsorship.txBytes,
  signature: [signedBySender.signature, sponsorship.signature],
});

console.log("txDigest:", exec.digest);
}
```

<https://sdk.mystenlabs.com/typescript/transaction-building/sponsored-transactions>

1 ガスレストラランザクション

②③スポンサーによる署名

ShinamiのSDKを使い、ガスの設定～署名を行なっています。

```
// 1) ガス無しTxKindを作る (時計アクセス)
const tx = new Transaction();
tx.moveCall({
  target:
    "0xfa0e78030bd16672174c2d6cc4cd5d1d1423d03c28a74909b2a148eda8bcca16::clock::access",
  arguments: [tx.object("0x6")],
});
const txKindBytes = await tx.build({
  client: node,
  onlyTransactionKind: true,
});
const txKindB64 = Buffer.from(txKindBytes).toString("base64");

// 2) スポンサー取得 (Auto-budget)
const sponsorship = await gas.sponsorTransaction({
  txKind: txKindB64,
  sender: keypair.toSuiAddress(),
});

// 3) 送信者署名
const signedBySender = await Transaction.from(sponsorship.txBytes).sign({
  signer: keypair,
});

// 4) 送信 (スポンサー署名 + 送信者署名)
const exec = await node.executeTransactionBlock({
  transactionBlock: sponsorship.txBytes,
  signature: [signedBySender.signature, sponsorship.signature],
});

console.log("txDigest:", exec.digest);
}
```

1 ガスレストラランザクション

④⑤ユーザーによる署名

スポンサーによりガスが設定された取引データに対して、署名しています。

```
// 1) ガス無しTxKindを作る (時計アクセス)
const tx = new Transaction();
tx.moveCall({
  target:
    "0xfa0e78030bd16672174c2d6cc4cd5d1d1423d03c28a74909b2a148eda8bcca16::clock::access",
  arguments: [tx.object("0x6")],
});
const txKindBytes = await tx.build({
  client: node,
  onlyTransactionKind: true,
});
const txKindB64 = Buffer.from(txKindBytes).toString("base64");

// 2) スポンサー取得 (Auto-budget)
const sponsorship = await gas.sponsorTransaction({
  txKind: txKindB64,
  sender: keypair.toSuiAddress(),
});

// 3) 送信者署名
const signedBySender = await Transaction.from(sponsorship.txBytes).sign({
  signer: keypair,
});

// 4) 送信 (スポンサー署名 + 送信者署名)
const exec = await node.executeTransactionBlock({
  transactionBlock: sponsorship.txBytes,
  signature: [signedBySender.signature, sponsorship.signature],
});

console.log("txDigest:", exec.digest);
}
```

1 ガスレストラランザクション

④⑤ユーザーによる署名

スポンサーによりガスが設定された取引データに対して、署名しています。

```
// 1) ガス無しTxKindを作る (時計アクセス)
const tx = new Transaction();
tx.moveCall({
  target:
    "0xfa0e78030bd16672174c2d6cc4cd5d1d1423d03c28a74909b2a148eda8bcca16::clock::access",
  arguments: [tx.object("0x6")],
});
const txKindBytes = await tx.build({
  client: node,
  onlyTransactionKind: true,
});
const txKindB64 = Buffer.from(txKindBytes).toString("base64");

// 2) スポンサー取得 (Auto-budget)
const sponsorship = await gas.sponsorTransaction({
  txKind: txKindB64,
  sender: keypair.toSuiAddress(),
});

// 3) 送信者署名
const signedBySender = await Transaction.from(sponsorship.txBytes).sign({
  signer: keypair,
});

// 4) 送信 (スポンサー署名 + 送信者署名)
const exec = await node.executeTransactionBlock({
  transactionBlock: sponsorship.txBytes,
  signature: [signedBySender.signature, sponsorship.signature],
});

console.log("txDigest:", exec.digest);
}
```

1 ガスレストラランザクション

⑤ トランザクションの実行

ユーザーとスポンサーの署名をつけて実行しています。

```
// 1) ガス無しTxKindを作る (時計アクセス)
const tx = new Transaction();
tx.moveCall({
  target:
    "0xfa0e78030bd16672174c2d6cc4cd5d1d1423d03c28a74909b2a148eda8bcca16::clock::access",
  arguments: [tx.object("0x6")],
});
const txKindBytes = await tx.build({
  client: node,
  onlyTransactionKind: true,
});
const txKindB64 = Buffer.from(txKindBytes).toString("base64");

// 2) スポンサー取得 (Auto-budget)
const sponsorship = await gas.sponsorTransaction({
  txKind: txKindB64,
  sender: keypair.toSuiAddress(),
});

// 3) 送信者署名
const signedBySender = await Transaction.from(sponsorship.txBytes).sign({
  signer: keypair,
});

// 4) 送信 (スポンサー署名 + 送信者署名)
const exec = await node.executeTransactionBlock({
  transactionBlock: sponsorship.txBytes,
  signature: [signedBySender.signature, sponsorship.signature],
});

console.log("txDigest:", exec.digest);
}
```

Suiでのオラクル利用

利用可能なオラクル

- Chainlink
- Band Protocol
- Mysten Labs製のsimple oracleとmeta oracle

開発者向けガイド

- 天気オラクルの実装例
<https://docs.sui.io/guides/developer/app-examples/weather-oracle#initialize-the-project>
- Moveオラクルのサンプルコード
<https://github.com/pentagonxyz/move-oracles>

2 オラクル

Sui Weather Oracleは、かなりの文量です。。

[Home](#) > [App Examples](#) > [Oracles](#) > [Sui Weather Oracle](#)

Sui Weather Oracle

This guide demonstrates writing a module (smart contract) in Move, deploying it on Devnet, and adding a backend, which fetches the weather data from the OpenWeather API every 10 minutes and updates the weather conditions for each city. The dApp created in this guide is called Sui Weather Oracle and it provides real-time weather data for over 1,000 locations around the world.

You can access and use the weather data from the OpenWeather API for various applications, such as randomness, betting, gaming, insurance, travel, education, or research. You can also mint a weather NFT based on the weather data of a city, using the `mint` function of the Sui Weather Oracle smart contract.

Prerequisites

- ☒ Install the latest version of Sui.
- ☒ Set up your Sui account and CLI environment.

> Create Sui account and setup CLI environment
- ☒ Obtain test tokens.

> How to obtain tokens

Move smart contract

<https://docs.sui.io/guides/developer/app-examples/weather-oracle>

2 オラクル

Sui Weather Oracleは、かなりの文量です。。

```
weather.move

/// Define a capability for the admin of the oracle.
public struct AdminCap has key, store { id: UID }

/// // Define a one-time witness to create the `Publisher` of the oracle.
public struct WEATHER has drop {}

// Define a struct for the weather oracle
public struct WeatherOracle has key {
  id: UID,
  /// The address of the oracle.
  address: address,
  /// The name of the oracle.
  name: String,
  /// The description of the oracle.
  description: String,
}

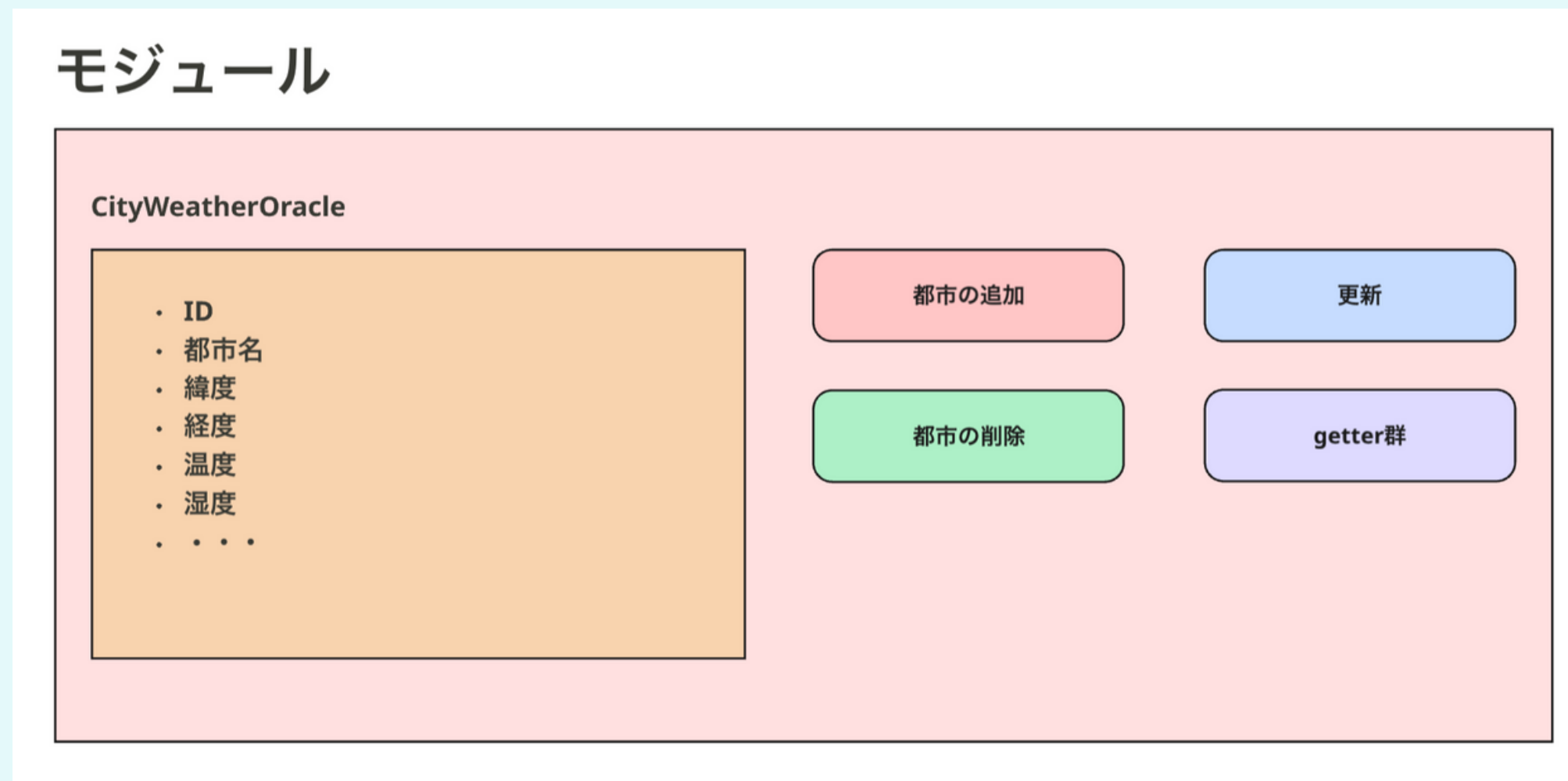
public struct CityWeatherOracle has key, store {
  id: UID,
  geoname_id: u32, // The unique identifier of the city
  name: String, // The name of the city
  country: String, // The country of the city
  latitude: u32, // The latitude of the city in degrees
  positive_latitude: bool, // Whether the latitude is positive (north) or negative (south)
  longitude: u32, // The longitude of the city in degrees
  positive_longitude: bool, // Whether the longitude is positive (east) or negative (west)
  weather_id: u16, // The weather condition code
  temp: u32, // The temperature in kelvin
  pressure: u32, // The atmospheric pressure in hPa
  humidity: u8, // The humidity percentage
  visibility: u16, // The visibility in meters
  wind_speed: u16, // The wind speed in meters per second
  wind_deg: u16, // The wind direction in degrees
  wind_gust: Option<u16>, // The wind gust in meters per second (optional)
  cloud_pct: u8, // The cloudiness percentage
}
```

<https://docs.sui.io/guides/developer/app-examples/weather-oracle>

2 オラクル

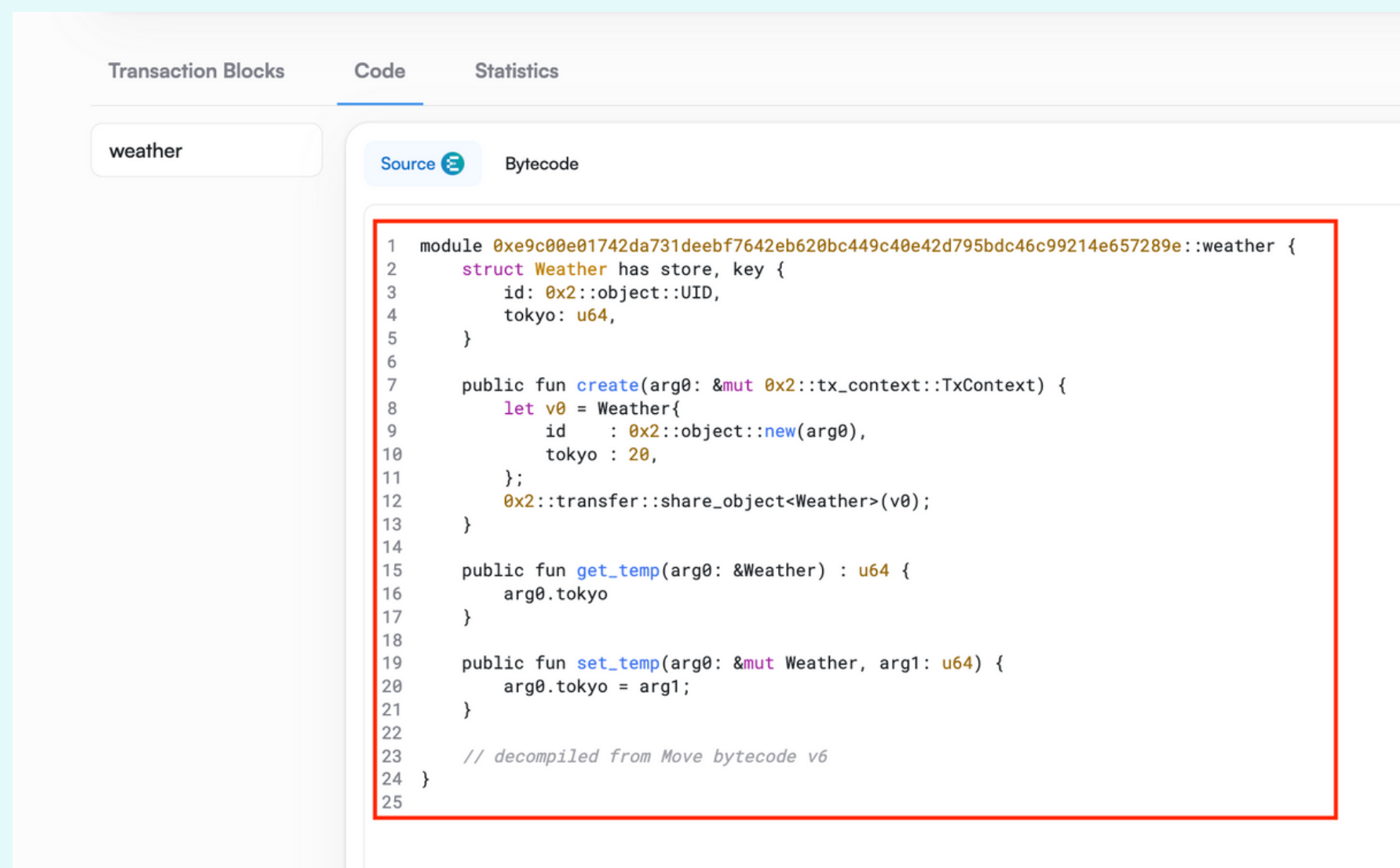
結局はただのモジュールです。**バックエンドで定期的に値を更新**します。

もちろんオラクルなので、**更新者が1人なのか分散した平均か**、また**権限**も重要にはなります。



2 オラクル

そのため、今回は、外部ですでに作られているパッケージの値を読み取り処理を行う方法を学びましょう。



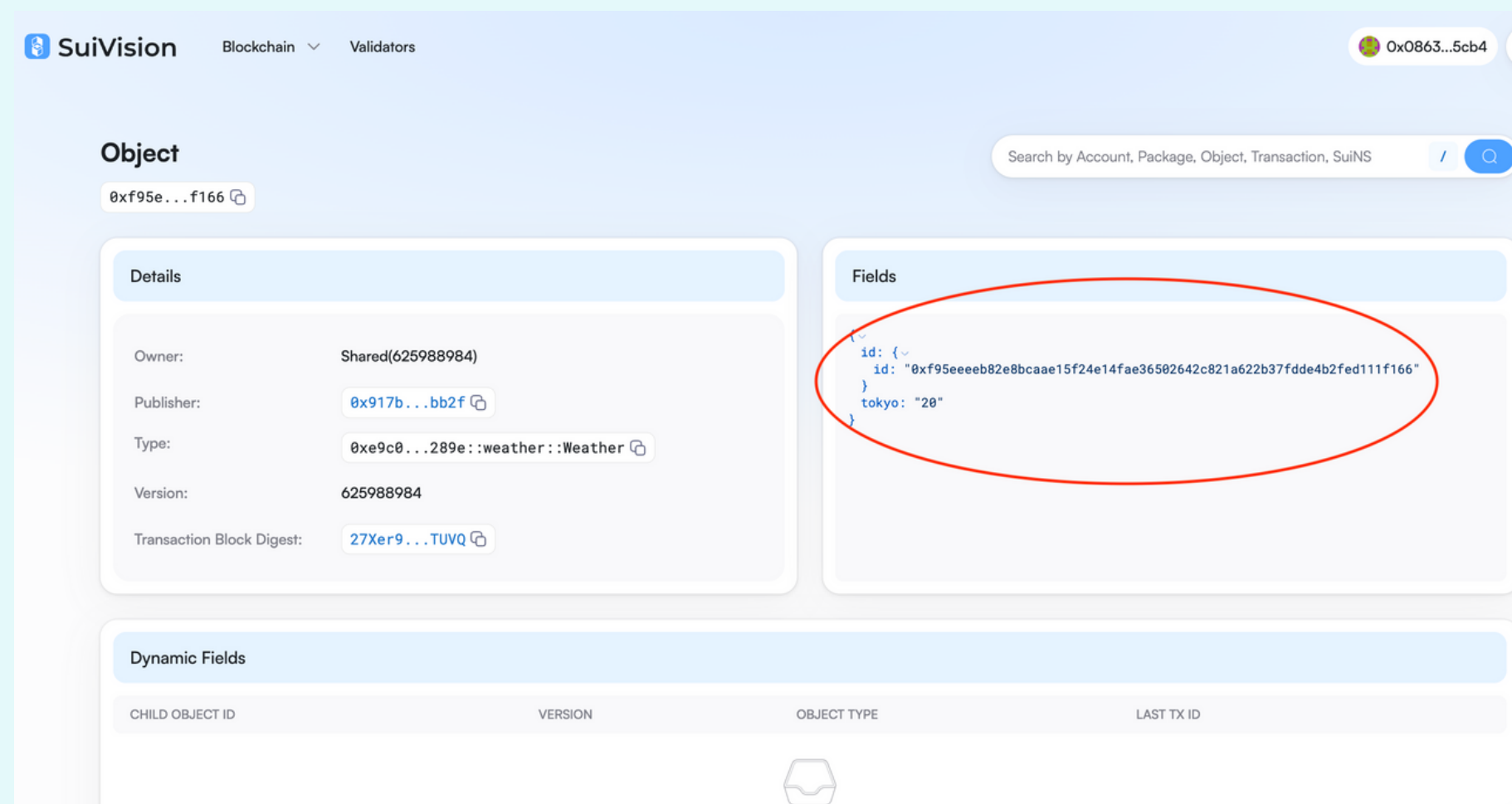
The screenshot shows a web interface for viewing Move code. On the left, there's a sidebar with a tab labeled 'weather'. The main area has two tabs: 'Source' (selected) and 'Bytecode'. The 'Source' tab displays the following code:

```
1 module 0xe9c00e01742da731deebf7642eb620bc449c40e42d795bdc46c99214e657289e::weather {  
2     struct Weather has store, key {  
3         id: 0x2::object::UID,  
4         tokyo: u64,  
5     }  
6  
7     public fun create(arg0: &mut 0x2::tx_context::TxContext) {  
8         let v0 = Weather{  
9             id : 0x2::object::new(arg0),  
10            tokyo : 20,  
11        };  
12        0x2::transfer::share_object<Weather>(v0);  
13    }  
14  
15    public fun get_temp(arg0: &Weather) : u64 {  
16        arg0.tokyo  
17    }  
18  
19    public fun set_temp(arg0: &mut Weather, arg1: u64) {  
20        arg0.tokyo = arg1;  
21    }  
22  
23    // decompiled from Move bytecode v6  
24 }  
25
```

<https://testnet.suivision.xyz/package/0xe9c00e01742da731deebf7642eb620bc449c40e42d795bdc46c99214e657289e?tab=Code>

2 オラクル

こちらがパッケージから作られたオブジェクトです。
東京が20度であることがわかります。
これを取得し、**20度以上なら〇〇する**という処理を作りましょう。



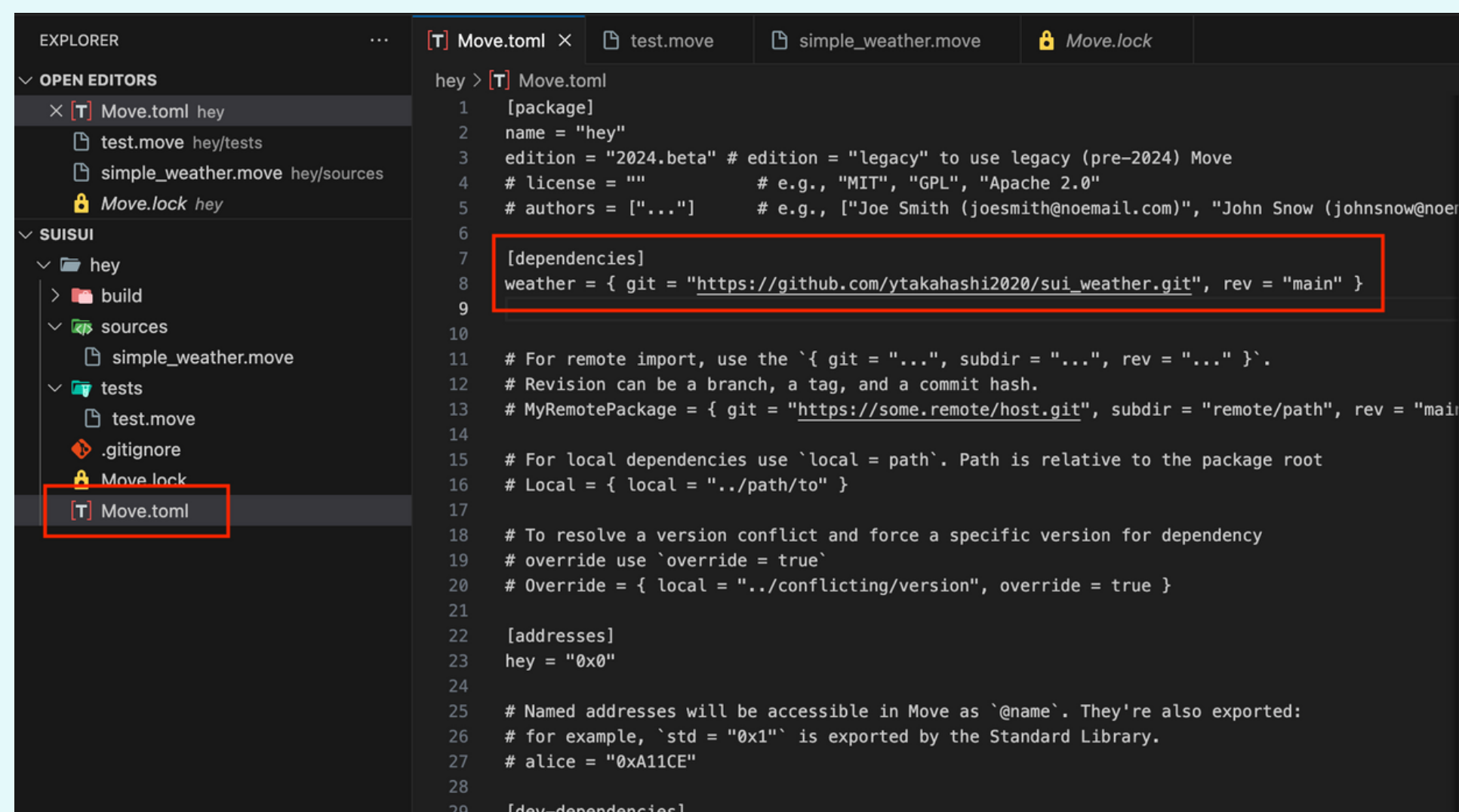
<https://testnet.suivision.xyz/object/0xf95eeeb82e8bcaae15f24e14fae36502642c821a622b37fdde4b2fed111f166>

2 オラクル

最も大事なのが、こちらです。

Move.tomlでパッケージのコードが格納されているGitを指定します。

https://github.com/ytakahashi2020/sui_weather



```
EXPLORER
  OPEN EDITORS
    Move.toml hey
    test.move hey/tests
    simple_weather.move hey/sources
    Move.lock hey
  SUI SUI
    hey
      build
      sources
        simple_weather.move
      tests
        test.move
        .gitignore
        Move.lock
        Move.toml
  Move.toml

hey > [T] Move.toml
1  [package]
2  name = "hey"
3  edition = "2024.beta" # edition = "legacy" to use legacy (pre-2024) Move
4  # license = ""          # e.g., "MIT", "GPL", "Apache 2.0"
5  # authors = ["..."]     # e.g., ["Joe Smith (joesmith@noemail.com)", "John Snow (johnsnow@noemail.com)"]
6
7  [dependencies]
8  weather = { git = "https://github.com/ytakahashi2020/sui_weather.git", rev = "main" }
9
10
11 # For remote import, use the `{ git = "...", subdir = "...", rev = "..." }`.
12 # Revision can be a branch, a tag, and a commit hash.
13 # MyRemotePackage = { git = "https://some.remote/host.git", subdir = "remote/path", rev = "main" }
14
15 # For local dependencies use `local = path`. Path is relative to the package root
16 # Local = { local = "../path/to" }
17
18 # To resolve a version conflict and force a specific version for dependency
19 # override use `override = true`
20 # Override = { local = "../conflicting/version", override = true }
21
22 [addresses]
23 hey = "0x0"
24
25 # Named addresses will be accessible in Move as `@name`. They're also exported:
26 # for example, `std = "0x1"` is exported by the Standard Library.
27 # alice = "0xA11CE"
28
29 [dev-dependencies]
```

https://github.com/ytakahashi2020/sui_oracle

2 オラクル

では、作ってみましょう。

ここでは、20度以上の場合、**イベントを発火するモジュール**を作ります。

まずは構造体を作ります。

```
hey > sources > simple_weather copy.move > ...  
1  module hey::weather_alert;  
2  
3  use std::string;  
4  use sui::event;  
5  
6  /// イベント構造体  
7  public struct HotEvent has copy, drop, store {  
8      message: string::String,  
9      temperature: u64,  
10 }  
11  
12
```

2 オラクル

次に、weatherパッケージのweatherモジュールの使用を宣言します。

```
hey > sources > simple_weather copy.move > ...
1  module hey::weather_alert;
2
3  use std::string;
4  use sui::event;
5  use weather::weather;
6
7  /// イベント構造体
8  public struct HotEvent has copy, drop, store {
9      message: string::String,
10     temperature: u64,
11 }
12
```

Owner: Immutable

Transaction Blocks Code Statistics

weather

Source Bytecode

```
1 module 0xe9c00e01742da731deebf7642eb620bc449c40e42d7
2     struct Weather has store, key {
3         id: 0x2::object::UID,
4         tokyo: u64,
5     }
6
7     public fun create(arg0: &mut 0x2::tx_context::Tx
8         let v0 = Weather{
9             id : 0x2::object::new(arg0),
10             tokyo : 20,
```

2 オラクル

次に、weatherモジュールのget_tempから値を取得します。

```
hey > sources > simple_weather.move > {} simple_weather > check_and_emit
1  module hey::simple_weather;
2
3  use std::string;
4  use sui::event;
5  use weather::weather;
6
7  /// イベント構造体
8  public struct HotEvent has copy, drop, store {
9      message: string::String,
10     temperature: u64,
11 }
12
13 /// Weatherオブジェクトを読み取って条件チェック
14 public fun check_and_emit(weather_obj: &weather::Weather, _ctx: &mut TxContext) {
15     let temp = weather::get_temp(weather_obj);
16
17 }
18
19
```

Transaction Blocks Code Statistics

weather

Source Bytecode

```
1 module 0xe9c00e01742da731deebf7642eb620bc449c40e42d795bdc46c99214e657289e::weather {
2     struct Weather has store, key {
3         id: 0x2::object::UID,
4         tokyo: u64,
5     }
6
7     public fun create(arg0: &mut 0x2::tx_context::TxContext) {
8         let v0 = Weather{
9             id : 0x2::object::new(arg0),
10            tokyo : 20,
11        };
12        0x2::transfer::share_object<Weather>(v0);
13    }
14
15    public fun get_temp(arg0: &Weather) : u64 {
16        arg0.tokyo
17    }
18
19    public fun set_temp(arg0: &mut Weather, arg1: u64) {
20        arg0.tokyo = arg1;
21    }
22
23    // decompiled from Move bytecode v6
24 }
25
```

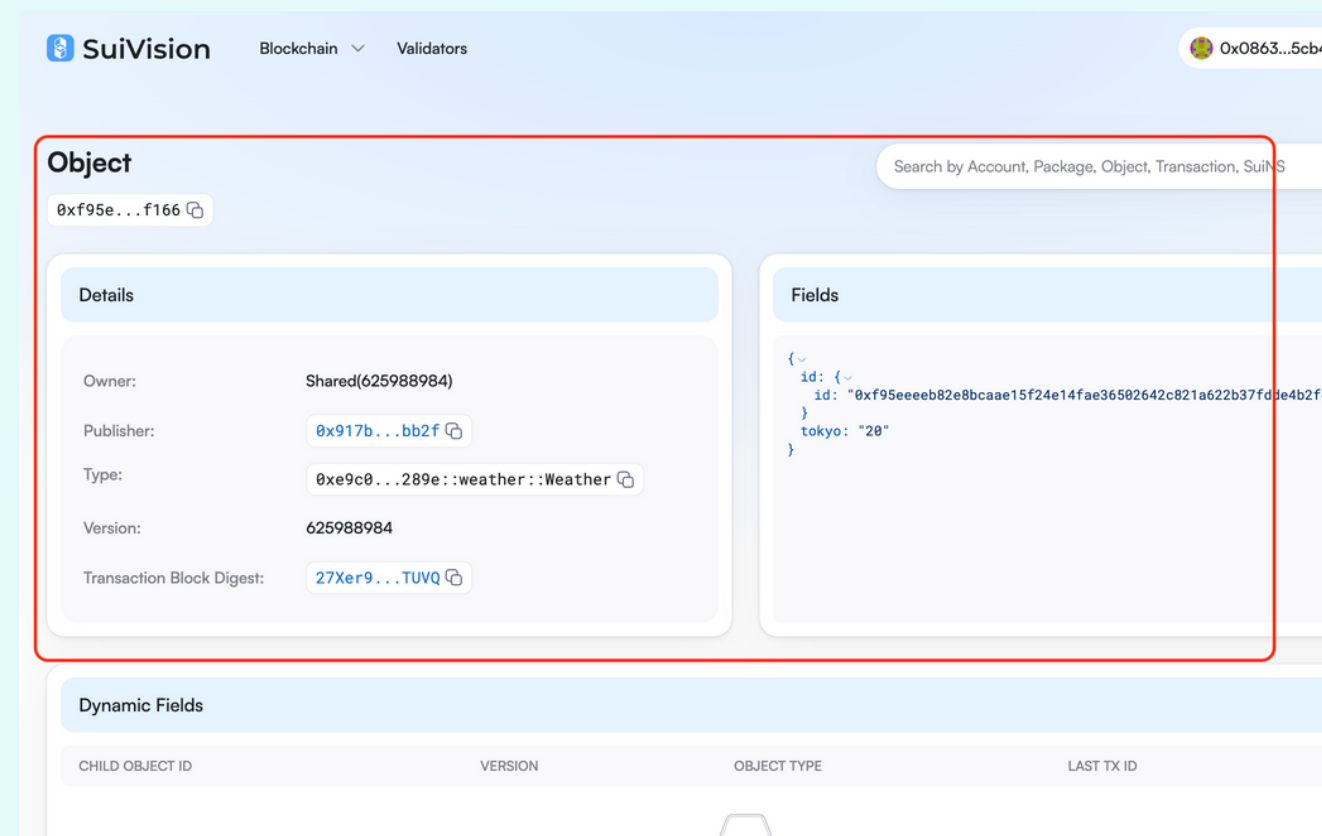
2 オラクル

最後にこのようにイベントを発火しています。

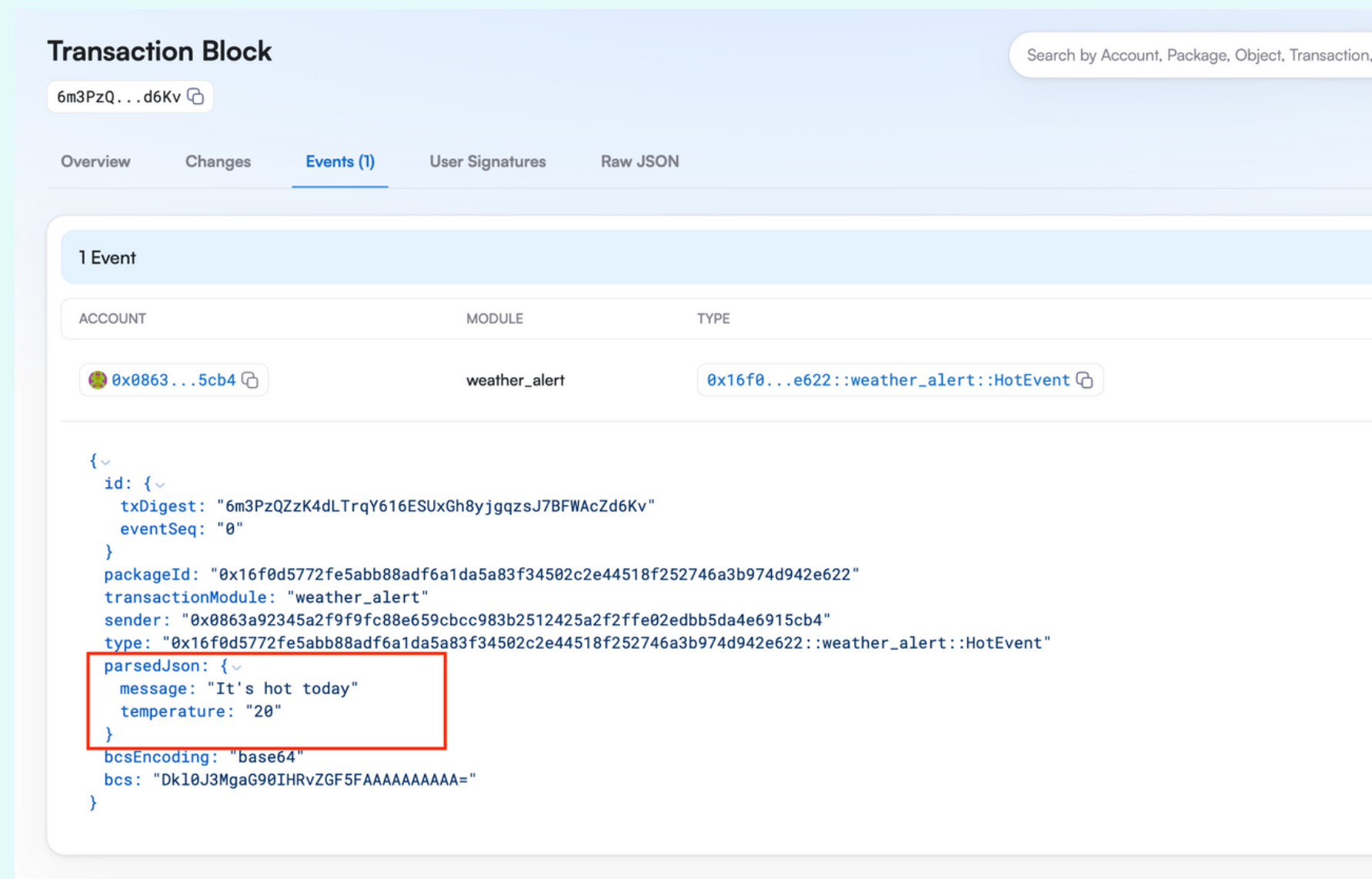
```
hey > sources > simple_weather.move > sui-move-analyzer > {} simple_weather > check_and_emit
1  module hey::simple_weather;
2
3  use std::string;
4  use sui::event;
5  use weather::weather;
6
7  /// イベント構造体
8  public struct HotEvent has copy, drop, store {
9      message: string::String,
10     temperature: u64,
11 }
12
13 /// Weatherオブジェクトを読み取って条件チェック
14 public fun check_and_emit(weather_obj: &weather::Weather, _ctx: &mut TxContext) {
15     let temp: u64 = weather::get_temp(weather: weather_obj);
16
17     if (temp >= 20) {
18         // 20度以上ならイベント発火
19         let message: String = b"It's hot today".to_string();
20         let event_data: HotEvent: HotEvent = HotEvent {
21             message: String,
22             temperature: u64: temp,
23         };
24         event::emit(event: event: event_data);
25     };
26 }
27
```

2 オラクル

できたパッケージの関数を実行すると、イベントが発火していることがわかります。



<https://testnet.suivision.xyz/object/0xf95e...f166>



<https://testnet.suivision.xyz/txblock/6m3PzQZzK4dLTrqY616ESUxGh8yjqzsJ7BFWAcZd6Kv?tab=Events>

SuiでのAIエージェント構築

エージェントロジック

- オンチェーンでのアクションはMoveスマートコントラクトで定義します。

オラクルと自動化

- データフィードを接続し、スポンサー付きトランザクションでプロセスを自動化します。

外部AIモデル

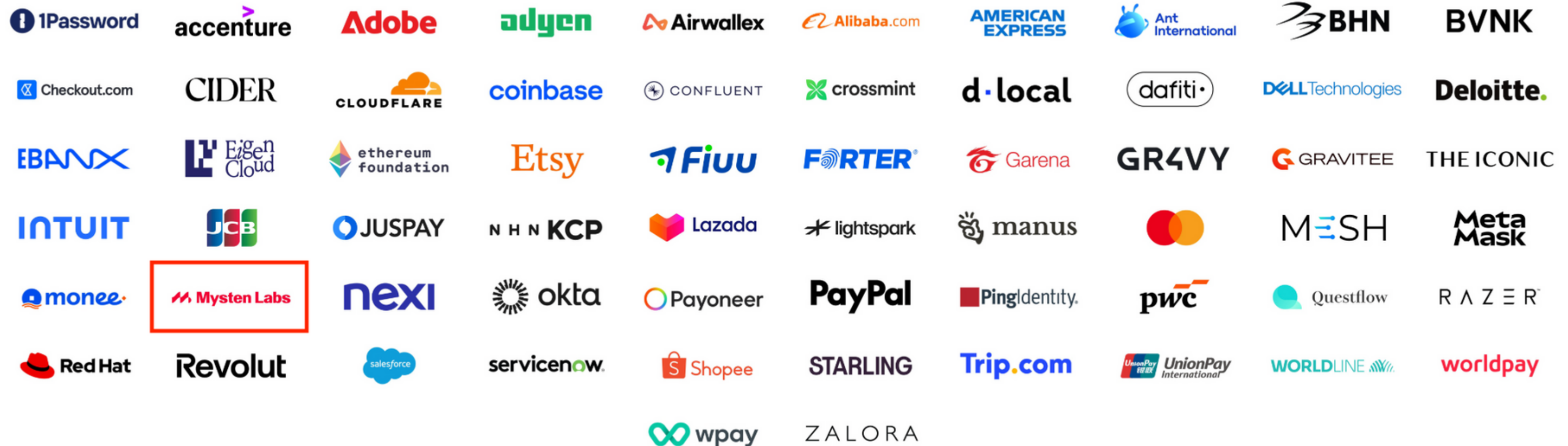
- 意思決定にオフチェーンのAIモデルを統合します。

セキュリティ

- Moveのケイパビリティとアクセス制御を活用します。



Partners contributing to the Agent Payments Protocol



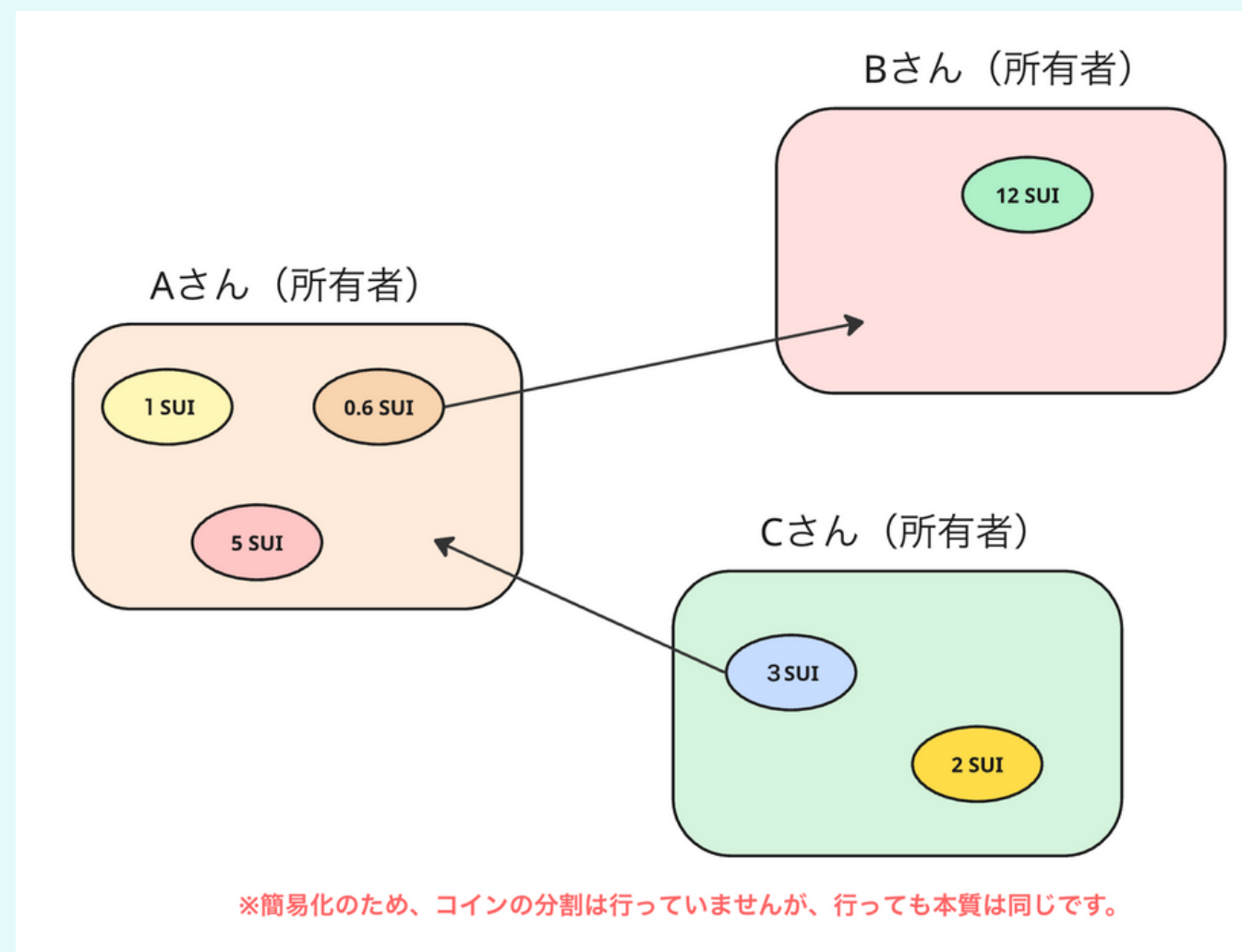
<https://cloud.google.com/blog/ja/products/ai-machine-learning/announcing-agents-to-payments-ap2-protocol>

(再掲)型からの派生②オブジェクトと所有権

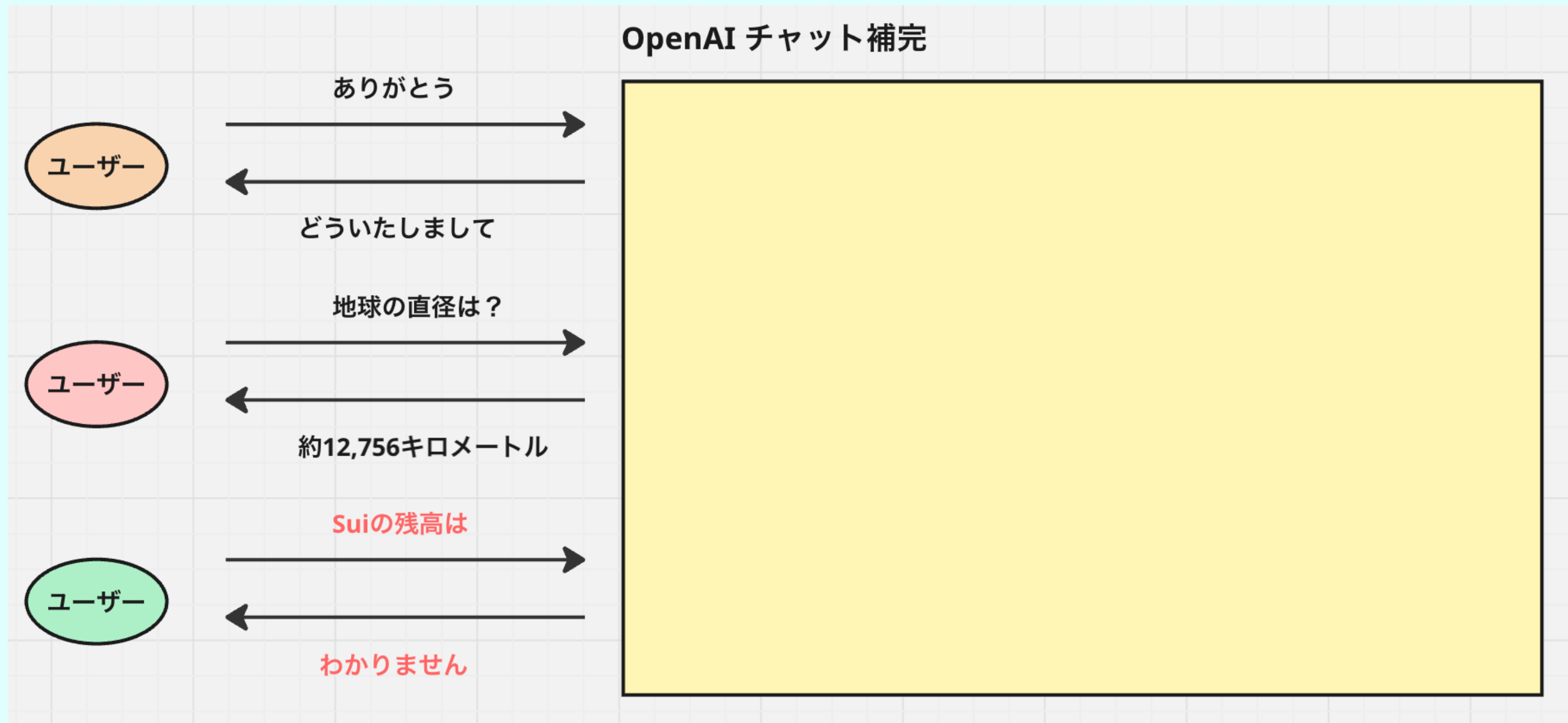
Suiの場合はオブジェクトを元にした所有権の設計です。

あるコインがアドレス所有である場合、**そのコインを移動できるのは所有者だけ**のため、**整合性の問題がそもそも発生しません。**

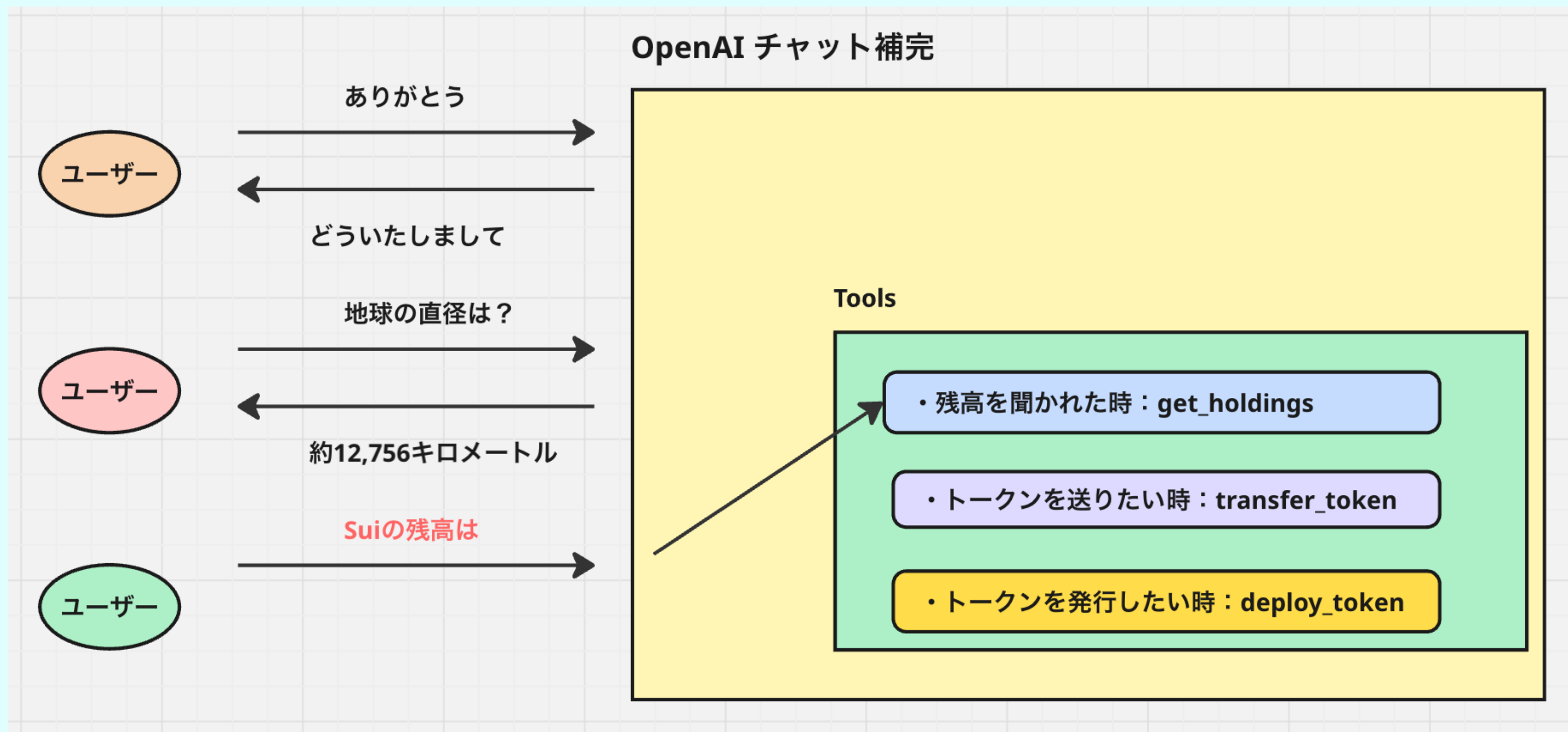
そのため**機械的に並列処理**を行うことができます。



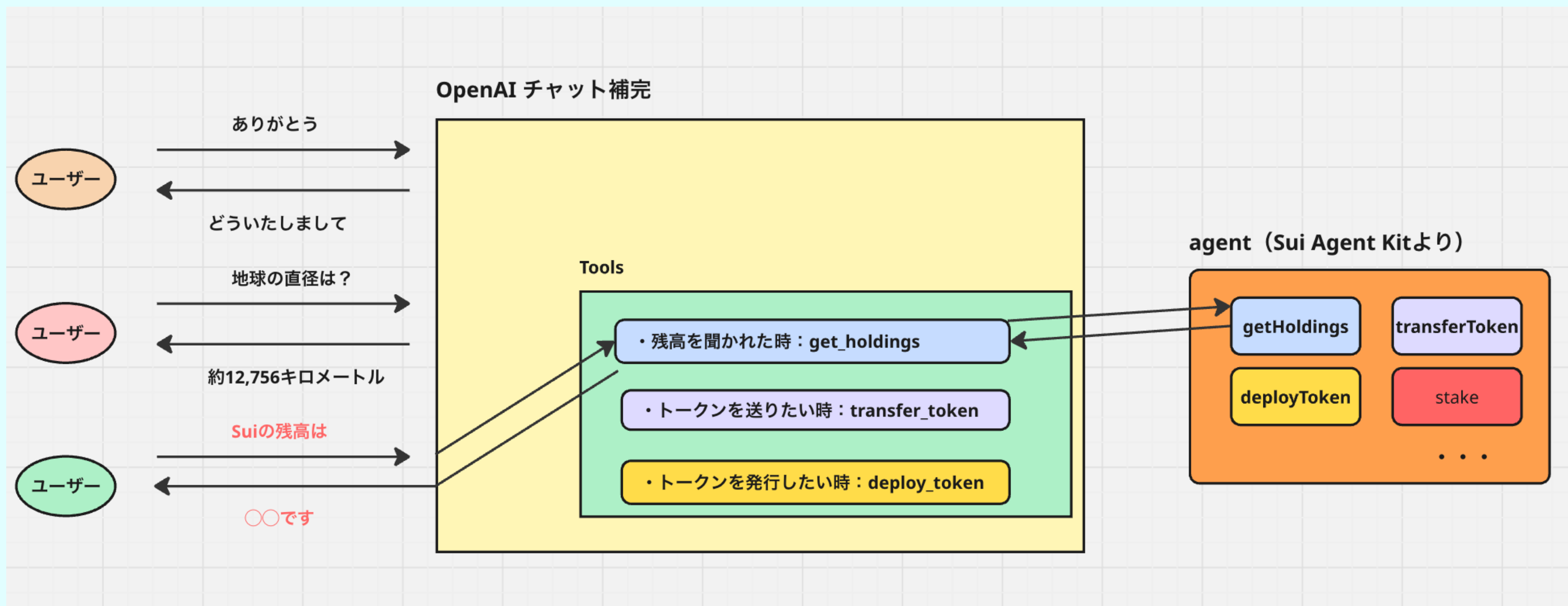
3 AIエージェント



3 AIエージェント

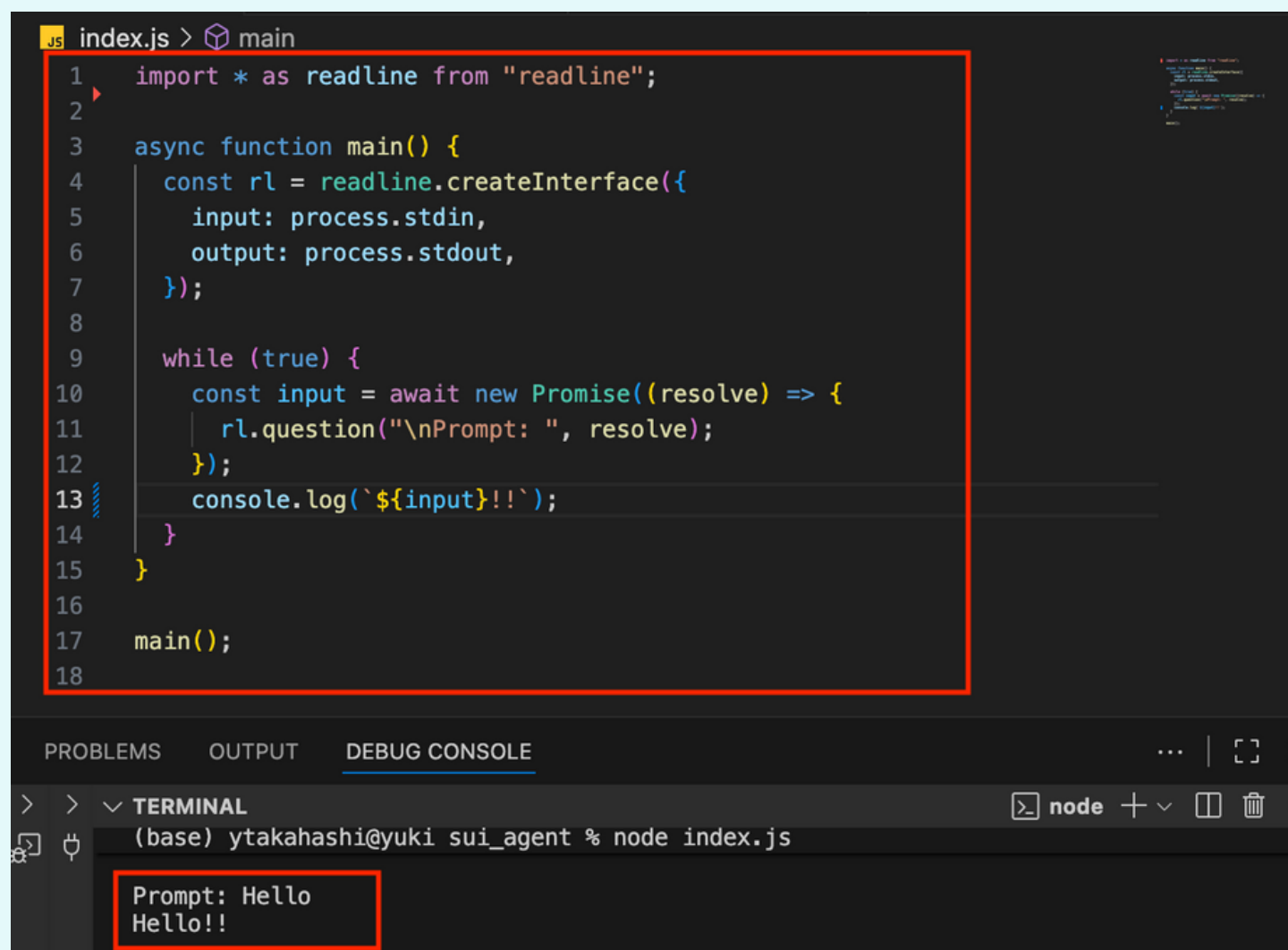


3 AIエージェント



3 AIエージェント

まずはreadlineを用いて、おうむ返しするコードを書いてみましょう。
npm init -yのあと、index.jsでファイルを作成しましょう。



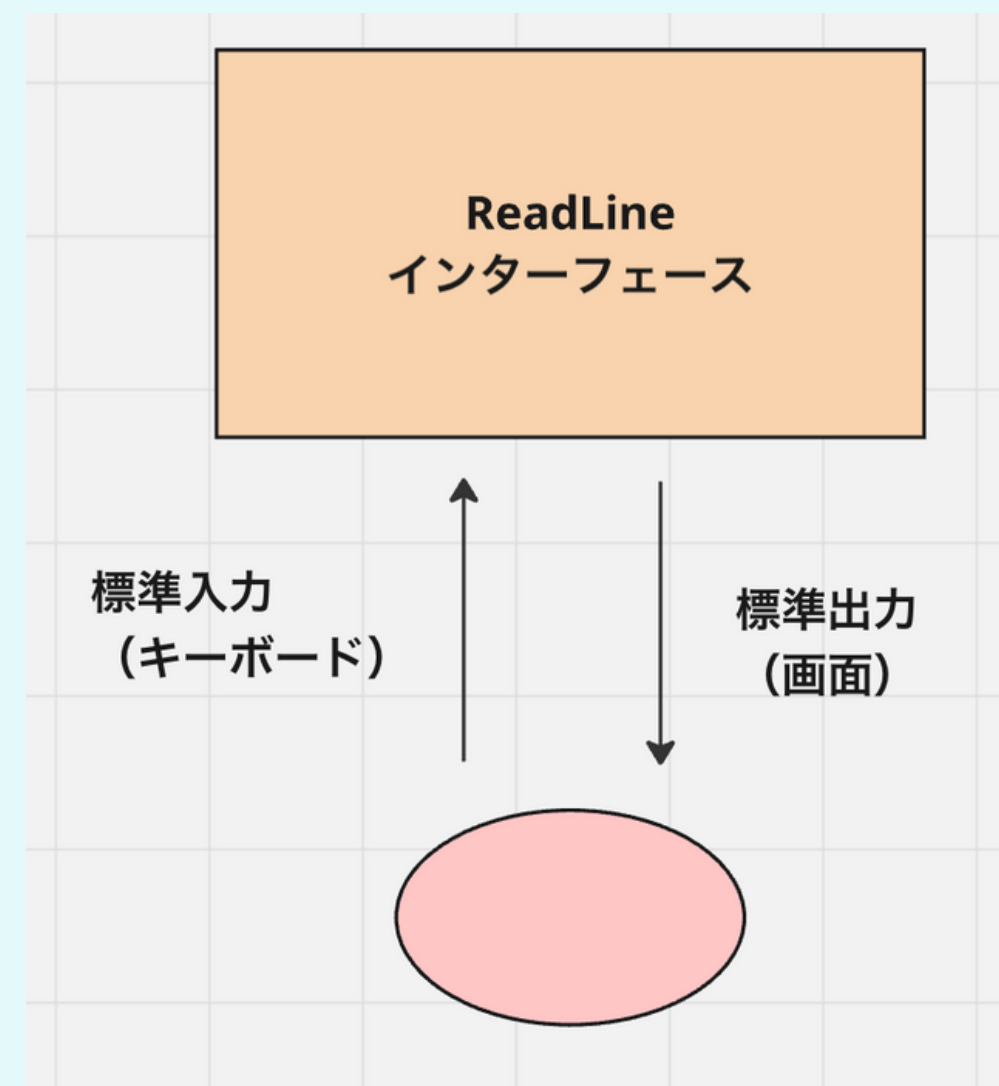
```
index.js > main
1 import * as readline from "readline";
2
3 async function main() {
4   const rl = readline.createInterface({
5     input: process.stdin,
6     output: process.stdout,
7   });
8
9   while (true) {
10    const input = await new Promise((resolve) => {
11      rl.question("\nPrompt: ", resolve);
12    });
13    console.log(`${input}!!`);
14  }
15 }
16
17 main();
18
```

PROBLEMS OUTPUT DEBUG CONSOLE

node

Prompt: Hello
Hello!!

https://github.com/ytakahashi2020/sui_agent



3 AIエージェント

dotenvを利用します。

npm i dotenv

```
JS index.js > main
1  import * as readline from "readline";
2  import * as dotenv from "dotenv";
3
4  dotenv.config();
5
6  async function main() {
7      const rl = readline.createInterface({
8          input: process.stdin,
9          output: process.stdout,
10     });
11
12     while (true) {
13         const input = await new Promise((resolve) => {
14             rl.question("\nPrompt: ", resolve);
15         });
16         console.log(`${input}!!`);
17     }
18 }
```

 ytakahashi2020 Update .env_example

Code

Blame

3 lines (3 loc) · 77 Bytes



1

OPENAI_API_KEY=

2

SUI_PRIVATE_KEY=

3

RPC_URL=https://fullnode.testnet.sui.io:443

https://github.com/ytakahashi2020/sui_agent/blob/main/.env_example

3 AIエージェント

openAIのインスタンスを作成します。

npm i openai

~/Desktop/sui_event/agent/index.js

```
1  import * as readline from "readline";
2  import * as dotenv from "dotenv";
3  import { OpenAI } from "openai/client.js";
4
5  dotenv.config();
6
7  const openai = new OpenAI({
8    apiKey: process.env.OPENAI_API_KEY,
9  });
10
11  const f_name = "get_holdings";
12  const tools = [
```

3 AIエージェント

inputを元にチャットを作しましょう。

```
index.js > ...
11  async function main() {
17  while (true) {
18    const input = await new Promise((resolve) => {
19      rl.question("\nPrompt: ", resolve);
20    });
21    const response = await openai.chat.completions.create({
22      model: "gpt-4o-mini",
23      messages: [
24        {
25          role: "system",
26          content: "You are helpful assistant",
27        },
28        {
29          role: "user",
30          content: input,
31        },
32      ],
33    });
34    console.log(response.choices[0].message.content);
35  }
36  }
37
38  main();
39
```

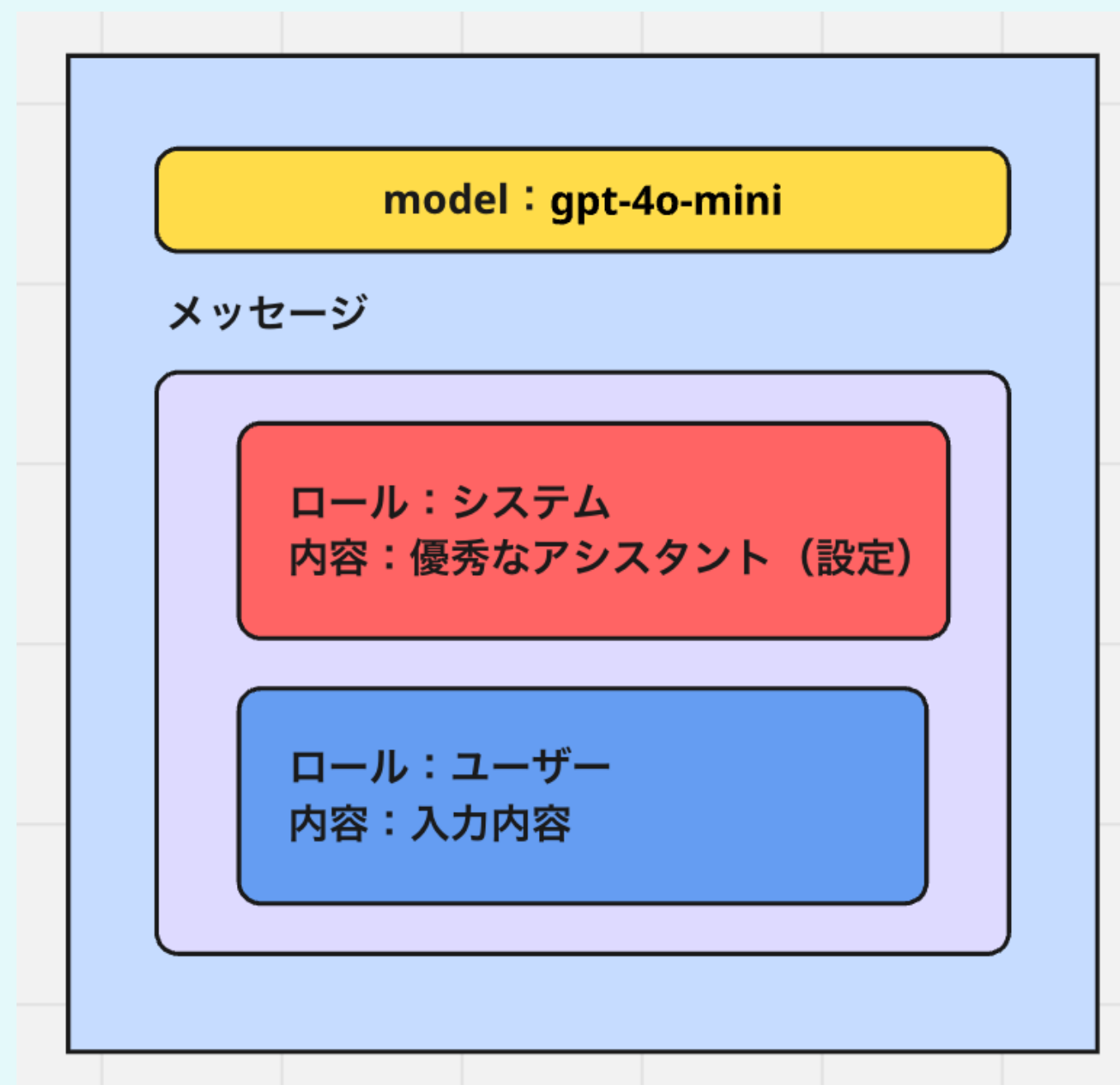
PROBLEMS OUTPUT DEBUG CONSOLE

> > > TERMINAL

(base) ytakahashi@yuki sui_agent % node index.js

Prompt: hi
Hello! How can I assist you today?

Prompt:



3 AIエージェント

inputを元にチャットを作しましょう。

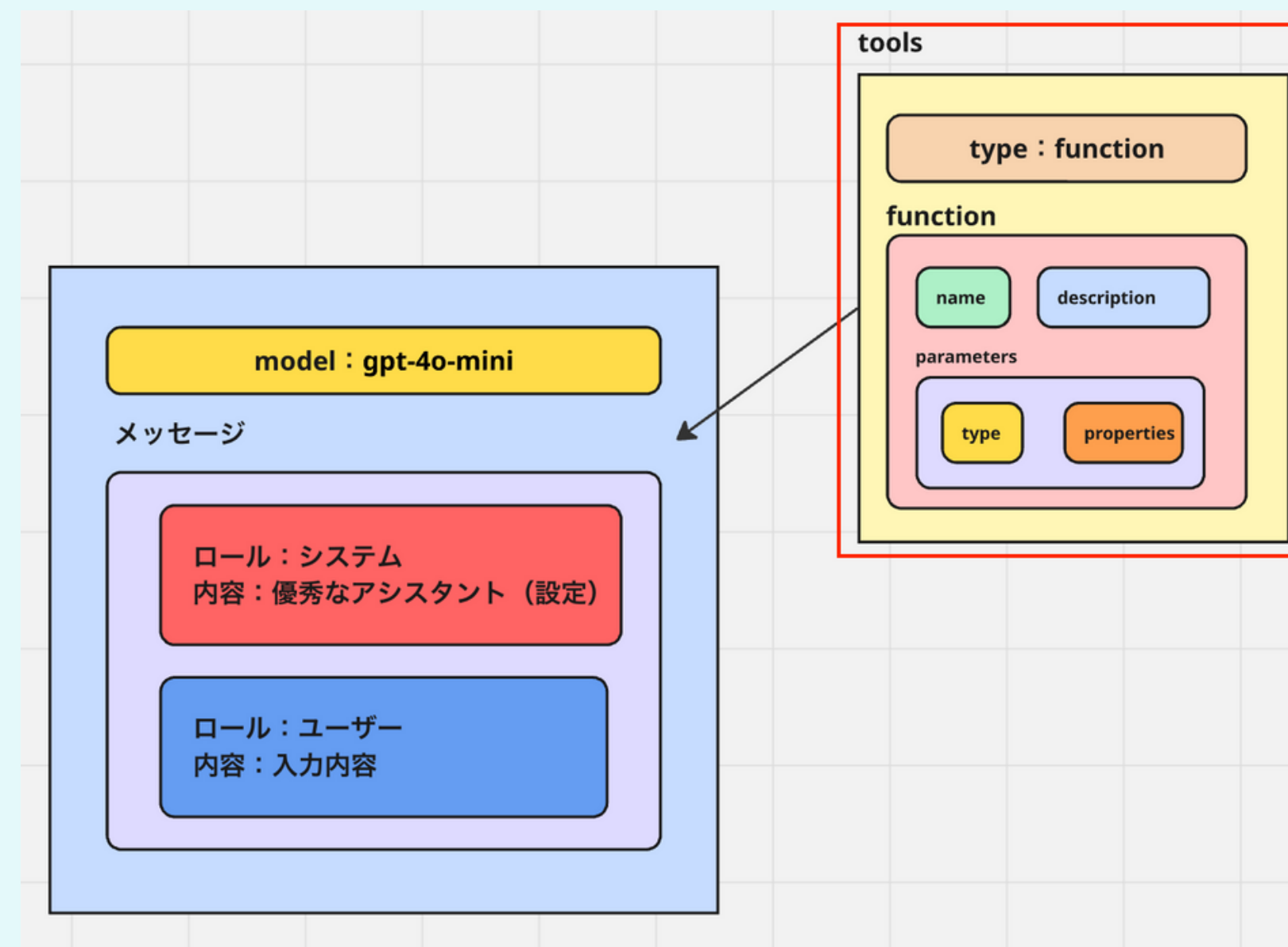
```
Prompt: hello
{
  id: 'chatcmp1-CHky0topf4M4Xktbnw9qzWXyqZqnf',
  object: 'chat.completion',
  created: 1758348152,
  model: 'gpt-4o-mini-2024-07-18',
  choices: [
    {
      index: 0,
      message: [Object],
      logprobs: null,
      finish_reason: 'stop'
    }
  ],
  usage: {
    prompt_tokens: 37,
    completion_tokens: 9,
    total_tokens: 46,
    prompt_tokens_details: { cached_tokens: 0, audio_tokens: 0 },
    completion_tokens_details: {
      reasoning_tokens: 0,
      audio_tokens: 0,
      accepted_prediction_tokens: 0,
      rejected_prediction_tokens: 0
    }
  }
}
```

```
Prompt: hello
{
  index: 0,
  message: {
    role: 'assistant',
    content: 'Hello! How can I assist you today?',
    refusal: null,
    annotations: []
  },
  logprobs: null,
  finish_reason: 'stop'
}
```

3 AIエージェント

inputを元にチャットを作しましょう。

```
index.js > main > [response]
10
11 const tools = [
12   {
13     type: "function",
14     function: {
15       name: "get_holdings",
16       description: "Get all token balances in the wallet",
17       parameters: {
18         type: "object",
19         properties: {},
20       },
21     },
22   },
23 ];
24
25 async function main() {
26   const rl = readline.createInterface({ ...
27   });
28
29   while (true) {
30     const input = await new Promise((resolve) => {
31       rl.question("\nPrompt: ", resolve);
32     });
33
34     const response = await openai.chat.completions.create({
35       model: "gpt-4o-mini",
36       messages: [
37         {
38           role: "system",
39           content: "You are helpful assistant",
40         },
41         {
42           role: "user",
43           content: input,
44         },
45       ],
46       tools: tools,
47       tool_choice: "auto",
48     });
49   }
```



3 AIエージェント

SuiAgentKitを作成します。
npm i @getnimbus/sui-agent-kit

```
3 import OpenAI from "openai";
4 import { SuiAgentKit } from "@getnimbus/sui-agent-kit";
5
6 dotenv.config();
7
8 const openai = new OpenAI({
9   apiKey: process.env.OPENAI_API_KEY,
10 });
11
12 const tools = [
13   {
14     type: "function",
15     function: {
16       name: "get_holdings",
17       description: "Get all token balances in the wallet",
18       parameters: {
19         type: "object",
20         properties: {},
21       },
22     },
23   },
24 ];
25
26 const agent = new SuiAgentKit(
27   process.env.SUI_PRIVATE_KEY,
28   process.env.RPC_URL,
29   {
30     OPENAI_API_KEY: process.env.OPENAI_API_KEY,
31   }
32 );
33
34 async function main() {
```

Token Operations

Check Holding Asset

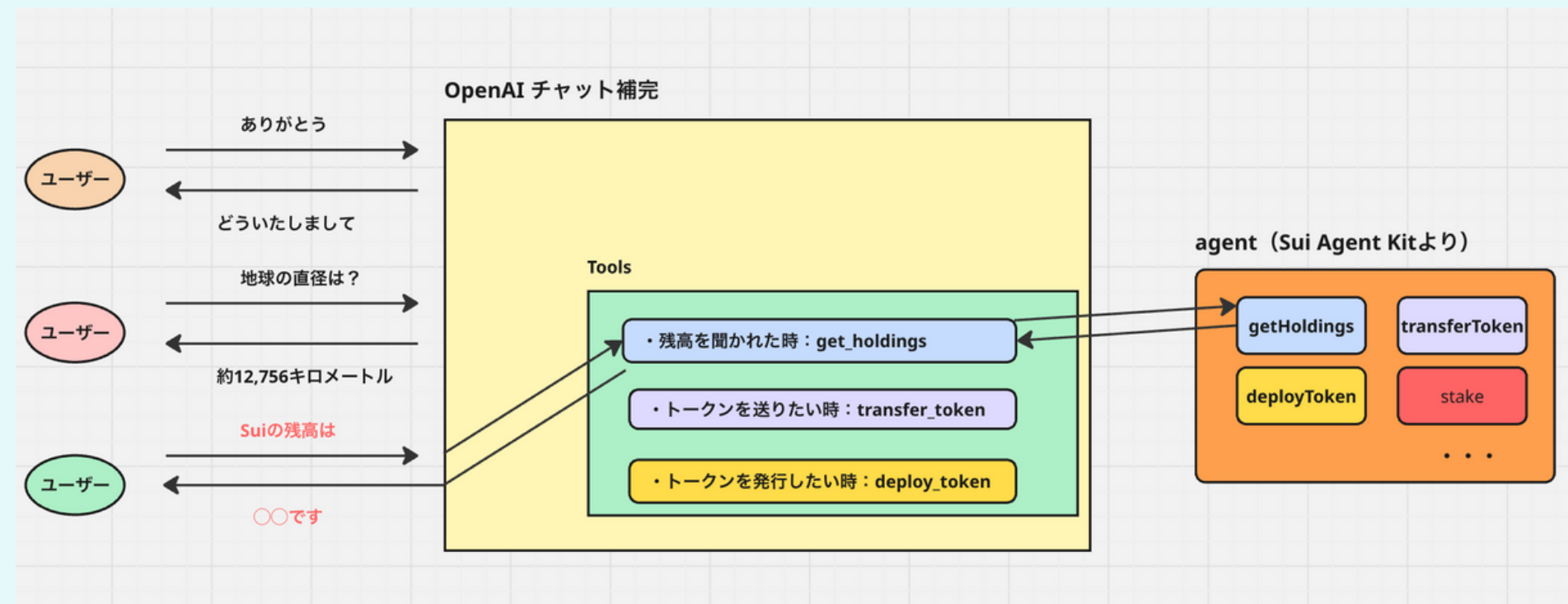
Learn how to check your wallet

The toolkit provides method:

- **getWalletHolding**: Check balances your own wallet

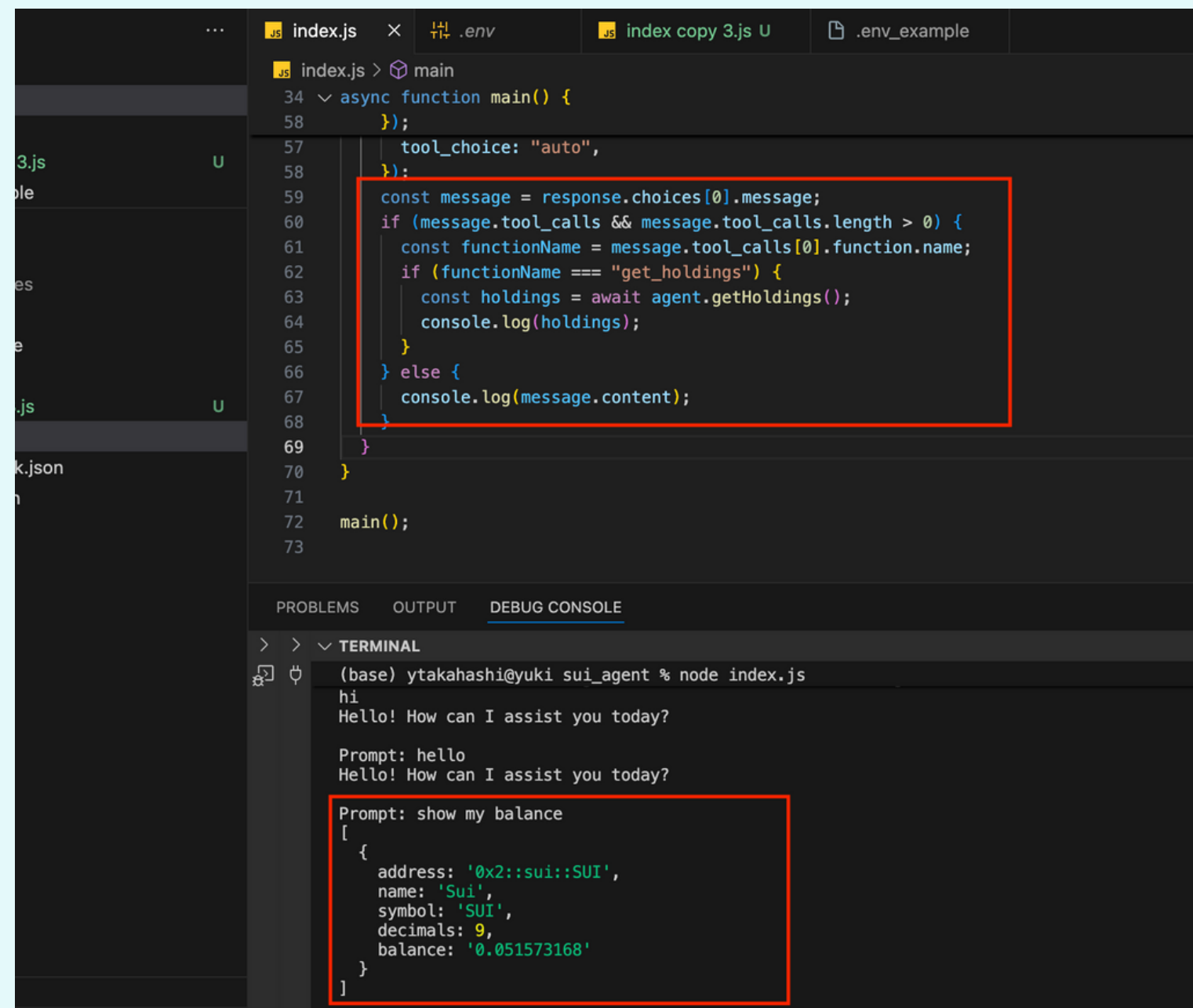
Usage

```
// Check your wallet's asset
const balance = await agent.getWalletHolding();
```



3 AIエージェント

最後に分岐させましょう。



```
index.js > main
34  async function main() {
58  };
57  tool_choice: "auto",
58  });
59  const message = response.choices[0].message;
60  if (message.tool_calls && message.tool_calls.length > 0) {
61    const functionName = message.tool_calls[0].function.name;
62    if (functionName === "get_holdings") {
63      const holdings = await agent.getHoldings();
64      console.log(holdings);
65    }
66  } else {
67    console.log(message.content);
68  }
69 }
70 }
71
72 main();
73
```

```
(base) ytakahashi@yuki sui_agent % node index.js
hi
Hello! How can I assist you today?

Prompt: hello
Hello! How can I assist you today?

Prompt: show my balance
[
  {
    address: '0x2::sui::SUI',
    name: 'Sui',
    symbol: 'SUI',
    decimals: 9,
    balance: '0.051573168'
  }
]
```