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.

This guide assumes you have installed Sui and understand Sui fundamentals.

As with all Sui dApps, a Move package on chain powers the logic of Sui Weather Oracle. The following instruction walks you through creating and publishing the module.

Before you get started, you must initialize a Move package. Open a terminal or console in the directory you want to store the example and run the following command to create an empty package with the name weather_oracle :

With that done, it's time to jump into some code. Create a new file in the sources directory with the name weather.move and populate the file with the following code:

There are few details to take note of in this code:

Next, add some more code to this module:

So far, you've set up the data structures within the module. Now, create a function that initializes a CityWeatherOracle and adds it as dynamic fields to the WeatherOracle object:

The add_city function is a public function that allows the owner of the AdminCap of the Sui Weather Oracle smart contract to add a new CityWeatherOracle . The function requires the admin to provide a capability object that proves their permission to add a city. The function also requires a mutable reference to the oracle object, which is the main object that stores the weather data on the blockchain. The function takes several parameters that describe the city, such as the geoname ID, name, country, latitude, longitude, and positive latitude and longitude. The function then creates a new city weather oracle object, which is a sub-object that stores and updates the weather data for a specific city. The function initializes the city weather oracle object with the parameters provided by the admin, and sets the weather data to be zero or none. The function then adds a new dynamic object field to the oracle object, using the geoname ID as the key and the city weather oracle object as the value. This way, the function adds a new city to the oracle, and makes it ready to receive and update the weather data from the backend service.

If you want to delete a city from the Sui Weather Oracle, call the remove_city function of the smart contract. The remove_city function allows the admin of the smart contract to remove a city from the oracle. The function requires the admin to provide a capability object that proves their permission to remove a city. The function also requires a mutable reference to the oracle object, which is the main object that stores and updates the weather data on the blockchain. The function takes the geoname ID of the city as a parameter, and deletes the city weather oracle object for the city. The function also removes the dynamic object field for the city from the oracle object. This way, the function deletes a city from the oracle, and frees up some storage space on the blockchain.

Now that you have implemented the add_city and remove_city functions, you can move on to the next step, which is to see how you can update the weather data for each city. The backend service fetches the weather data from the OpenWeather API every 10 minutes, and then passes the data to the update function of the Sui Weather Oracle smart contract. The update function takes the geoname ID and the new weather data of the city as parameters, and updates the city weather oracle object with the new data. This way, the weather data on the blockchain is always up to date and accurate.

You have defined the data structure of the Sui Weather Oracle smart contract, but you need some functions to access and manipulate the data. Now, add some helper functions that read and return the weather data for a WeatherOracle object. These functions allow you to get the weather data for a specific city in the oracle. These functions also allow you to format and display the weather data in a user-friendly way.

Finally, add an extra feature that allows anyone to mint a WeatherNFT with the current conditions of a city by passing the geoname ID. The mint function is a public function that allows anyone to mint a weather NFT based on the weather data of a city. The function takes the WeatherOracle shared object and the geoname ID of the city as parameters, and returns a new WeatherNFT object for the city. The WeatherNFT object is a unique and non-fungible token that represents the weather of the city at the time of minting. The WeatherNFT object has the same data as the CityWeatherOracle object, such as the geonameID , name , country , latitude , longitude , positive latitude and longitude , weather ID , temperature , pressure , humidity , visibility , wind speed , wind

degree , wind gust , clouds , and timestamp . The function creates the WeatherNFT object by borrowing a reference to the CityWeatherOracle object with the geoname ID as the key, and assigning a unique ID (UID) to the WeatherNFT object. The function then transfers the ownership of the WeatherNFT object to the sender of the transaction. This way, the function allows anyone to mint a weather NFT and own a digital representation of the weather of a city. You can use this feature to create your own collection of weather NFTs, or to use them for other applications that require verifiable and immutable weather data.

And with that, your weather.move code is complete.

See Publish a Package for a more detailed guide on publishing packages or Sui Client CLI for a complete reference of client commands in the Sui CLI.

Before publishing your code, you must first initialize the Sui Client CLI, if you haven't already. To do so, in a terminal or console at the root directory of the project enter sui client . If you receive the following response, complete the remaining instructions:

Enter y to proceed. You receive the following response:

Leave this blank (press Enter). You receive the following response:

Select 0 . Now you should have a Sui address set up.

Before being able to publish your package to Testnet, you need Testnet SUI tokens. To get some, visit the online faucet at https://faucet.sui.io/ . For other ways to get SUI in your Testnet account, see Get SUI Tokens .

Now that you have an account with some Testnet SUI, you can deploy your contracts. To publish your package, use the following command in the same terminal or console:

For the gas budget, use a standard value such as 20000000 .

You have successfully deployed the Sui Weather Oracle smart contract on the blockchain. Now, it's time to create an Express backend that can interact with it. The Express backend performs the following tasks:

The Express backend uses the Sui Typescript SDK , a TypeScript library that enables you to interact with the Sui blockchain and smart contracts. With the Sui Typescript SDK, you can connect to the Sui network, sign and submit transactions, and query the state of the smart contract. You also use the OpenWeather API to fetch the weather data for each city and update the smart contract every 10 minutes. Additionally, you can mint weather NFTs, if you want to explore that feature of the smart contract.

First, initialize your backend project. To do this, you need to follow these steps:

The code of init.ts does the following:

You have now initialized the WeatherOracle shared object. The next step is to learn how to update them every 10 minutes with the latest weather data from the OpenWeatherMap API.

The code in index.ts does the following:

Congratulations, you completed the Sui Weather Oracle tutorial. You can carry the lessons learned here forward when building your next Sui project.

Move smart contract

As with all Sui dApps, a Move package on chain powers the logic of Sui Weather Oracle. The following instruction walks you through creating and publishing the module.

Before you get started, you must initialize a Move package. Open a terminal or console in the directory you want to store the example and run the following command to create an empty package with the name weather_oracle :

With that done, it's time to jump into some code. Create a new file in the sources directory with the name weather.move and populate the file with the following code:

There are few details to take note of in this code:

Next, add some more code to this module:

So far, you've set up the data structures within the module. Now, create a function that initializes a CityWeatherOracle and adds it as dynamic fields to the WeatherOracle object:

The add_city function is a public function that allows the owner of the AdminCap of the Sui Weather Oracle smart contract to add a new CityWeatherOracle . The function requires the admin to provide a capability object that proves their permission to add a city. The function also requires a mutable reference to the oracle object, which is the main object that stores the weather data on the blockchain. The function takes several parameters that describe the city, such as the geoname ID, name, country, latitude, longitude, and positive latitude and longitude. The function then creates a new city weather oracle object, which is a sub-object that stores and updates the weather data for a specific city. The function initializes the city weather oracle object with the parameters provided by the admin, and sets the weather data to be zero or none. The function then adds a new dynamic object field to the oracle object, using the geoname ID as the key and the city weather oracle object as the value. This way, the function adds a new city to the oracle, and makes it ready to receive and update the weather data from the backend service.

If you want to delete a city from the Sui Weather Oracle, call the remove_city function of the smart contract. The remove_city function allows the admin of the smart contract to remove a city from the oracle. The function requires the admin to provide a capability object that proves their permission to remove a city. The function also requires a mutable reference to the oracle object, which is the main object that stores and updates the weather data on the blockchain. The function takes the geoname ID of the city as a parameter, and deletes the city weather oracle object for the city. The function also removes the dynamic object field for the city from the oracle object. This way, the function deletes a city from the oracle, and frees up some storage space on the blockchain.

Now that you have implemented the add_city and remove_city functions, you can move on to the next step, which is to see how you can update the weather data for each city. The backend service fetches the weather data from the OpenWeather API every 10 minutes, and then passes the data to the update function of the Sui Weather Oracle smart contract. The update function takes the geoname ID and the new weather data of the city as parameters, and updates the city weather oracle object with the new data. This way, the weather data on the blockchain is always up to date and accurate.

You have defined the data structure of the Sui Weather Oracle smart contract, but you need some functions to access and manipulate the data. Now, add some helper functions that read and return the weather data for a WeatherOracle object. These functions allow you to get the weather data for a specific city in the oracle. These functions also allow you to format and display the weather data in a user-friendly way.

Finally, add an extra feature that allows anyone to mint a WeatherNFT with the current conditions of a city by passing the geoname ID. The mint function is a public function that allows anyone to mint a weather NFT based on the weather data of a city. The function takes the WeatherOracle shared object and the geoname ID of the city as parameters, and returns a new WeatherNFT object for the city. The WeatherNFT object is a unique and non-fungible token that represents the weather of the city at the time of minting. The WeatherNFT object has the same data as the CityWeatherOracle object, such as the geonameID , name , country , latitude , longitude , positive latitude and longitude , weather ID , temperature , pressure , humidity , visibility , wind speed , wind degree , wind gust , clouds , and timestamp . The function creates the WeatherNFT object by borrowing a reference to the CityWeatherOracle object with the geoname ID as the key, and assigning a unique ID (UID) to the WeatherNFT object. The function then transfers the ownership of the WeatherNFT object to the sender of the transaction. This way, the function allows anyone to mint a weather NFT and own a digital representation of the weather of a city. You can use this feature to create your own collection of weather NFTs, or to use them for other applications that require verifiable and immutable weather data.

And with that, your weather.move code is complete.

See Publish a Package for a more detailed guide on publishing packages or Sui Client CLI for a complete reference of client commands in the Sui CLI.

Before publishing your code, you must first initialize the Sui Client CLI, if you haven't already. To do so, in a terminal or console at the root directory of the project enter sui client . If you receive the following response, complete the remaining instructions:

Enter y to proceed. You receive the following response:

Leave this blank (press Enter). You receive the following response:

Select 0 . Now you should have a Sui address set up.

Before being able to publish your package to Testnet, you need Testnet SUI tokens. To get some, visit the online faucet at https://faucet.sui.io/ . For other ways to get SUI in your Testnet account, see Get SUI Tokens .

Now that you have an account with some Testnet SUI, you can deploy your contracts. To publish your package, use the following command in the same terminal or console:

For the gas budget, use a standard value such as 20000000 .

You have successfully deployed the Sui Weather Oracle smart contract on the blockchain. Now, it's time to create an Express backend that can interact with it. The Express backend performs the following tasks:

The Express backend uses the Sui Typescript SDK , a TypeScript library that enables you to interact with the Sui blockchain and smart contracts. With the Sui Typescript SDK, you can connect to the Sui network, sign and submit transactions, and query the state of the smart contract. You also use the OpenWeather API to fetch the weather data for each city and update the smart contract every 10 minutes. Additionally, you can mint weather NFTs, if you want to explore that feature of the smart contract.

First, initialize your backend project. To do this, you need to follow these steps:

The code of init.ts does the following:

You have now initialized the WeatherOracle shared object. The next step is to learn how to update them every 10 minutes with the latest weather data from the OpenWeatherMap API.

The code in index.ts does the following:

Congratulations, you completed the Sui Weather Oracle tutorial. You can carry the lessons learned here forward when building your next Sui project.

Deployment

See Publish a Package for a more detailed guide on publishing packages or Sui Client CLI for a complete reference of client commands in the Sui CLI.

Before publishing your code, you must first initialize the Sui Client CLI, if you haven't already. To do so, in a terminal or console at the root directory of the project enter sui client . If you receive the following response, complete the remaining instructions:

Enter y to proceed. You receive the following response:

Leave this blank (press Enter). You receive the following response:

Select 0 . Now you should have a Sui address set up.

Before being able to publish your package to Testnet, you need Testnet SUI tokens. To get some, visit the online faucet at https://faucet.sui.io/ . For other ways to get SUI in your Testnet account, see Get SUI Tokens .

Now that you have an account with some Testnet SUI, you can deploy your contracts. To publish your package, use the following command in the same terminal or console:

For the gas budget, use a standard value such as 20000000 .

You have successfully deployed the Sui Weather Oracle smart contract on the blockchain. Now, it's time to create an Express backend that can interact with it. The Express backend performs the following tasks:

The Express backend uses the Sui Typescript SDK , a TypeScript library that enables you to interact with the Sui blockchain and smart contracts. With the Sui Typescript SDK, you can connect to the Sui network, sign and submit transactions, and query the state of the smart contract. You also use the OpenWeather API to fetch the weather data for each city and update the smart contract every 10 minutes. Additionally, you can mint weather NFTs, if you want to explore that feature of the smart contract.

First, initialize your backend project. To do this, you need to follow these steps:

The code of init.ts does the following:

You have now initialized the WeatherOracle shared object. The next step is to learn how to update them every 10 minutes with the latest weather data from the OpenWeatherMap API.

The code in index.ts does the following:

Congratulations, you completed the Sui Weather Oracle tutorial. You can carry the lessons learned here forward when building your next Sui project.

Backend

You have successfully deployed the Sui Weather Oracle smart contract on the blockchain. Now, it's time to create an Express backend that can interact with it. The Express backend performs the following tasks:

The Express backend uses the Sui Typescript SDK , a TypeScript library that enables you to interact with the Sui blockchain and smart contracts. With the Sui Typescript SDK, you can connect to the Sui network, sign and submit transactions, and query the state of the smart contract. You also use the OpenWeather API to fetch the weather data for each city and update the smart contract every 10 minutes. Additionally, you can mint weather NFTs, if you want to explore that feature of the smart contract.

First, initialize your backend project. To do this, you need to follow these steps:

The code of init.ts does the following:

You have now initialized the WeatherOracle shared object. The next step is to learn how to update them every 10 minutes with the latest weather data from the OpenWeatherMap API.

The code in index.ts does the following:

Congratulations, you completed the Sui Weather Oracle tutorial. You can carry the lessons learned here forward when building your next Sui project.