Write a Move Package

To begin, open a terminal or console at the location you plan to store your package. Use the sui move new command to create an empty Move package with the name my_first_package :

Running the previous command creates a directory with the name you provide (my_first_package in this case). The command populates the new directory with a skeleton Move project that consists of a sources directory and a Move.toml manifest file. Open the manifest with a text editor to review its contents:

The manifest file contents include available sections of the manifest and comments that provide additional information. In Move, you prepend the hash mark (#) to a line to denote a comment.

You have a package now but it doesn't do anything. To make your package useful, you must add logic contained in .move source files that define modules . The sui move new command creates a .move file in the sources directory that defaults to the same name as your project (my_first_package.move in this case). For the purpose of this guide, rename the file to example.move and open it with a text editor.

Populate the example.move file with the following code:

The comments in the preceding code highlight different parts of a typical Move source file.

Part 1: Imports - Code reuse is a necessity in modern programming. Move supports this concept with use aliases that allow your module to refer to types and functions declared in other modules. In this example, the module imports from object , transfer , and tx_context modules, but it does not need to do so explicitly, because the compiler provides these use statements by default. These modules are available to the package because the Move.toml file defines the Sui dependency (along with the sui named address) where they are defined.

Part 2: Struct declarations - Structs define types that a module can create or destroy. Struct definitions can include abilities provided with the has keyword. The structs in this example, for instance, have the key ability, which indicates that these structs are Sui objects that you can transfer between addresses. The store ability on the structs provides the ability to appear in other struct fields and be transferred freely.

Part 3: Module initializer - A special function that is invoked exactly once when the module publishes.

Part 4: Accessor functions - These functions allow the fields of the module's structs to be read from other modules.

After you save the file, you have a complete Move package.

Related links