Address-Owned Objects

An address-owned object is owned by a specific 32-byte address that is either an account address (derived from a particular signature scheme) or an object ID. An address-owned object is accessible only to its owner and no others.

As the owner (of the address that owns) an address-owned objects, you can transfer it to different addresses. Because only a single owner can access these objects, you can execute transactions that use only owned objects in parallel with other transactions that don't have any objects in common without needing to go through consensus.

You can use one of the following functions (defined in the transfer module) to create address-owned objects.

You should use the sui::transfer::transfer function if you are defining a custom transfer policy for the object. Use the sui::transfer::public_transfer function to create an address-owned object if the object has the store capability.

After you declare an object as address-owned, its ownership can change over the life of that object - either by adding it as a dynamic object field, transferring it to a different address, or making it immutable. Importantly though, after you create an object and set its ownership, it cannot be shared.

You can access address-owned objects in two different ways, depending on whether or not the address owner of the object corresponds to an object ID.

If the address owner of the object corresponds to an object ID, then you must access and dynamically authenticate it during the execution of the transaction using the mechanisms defined in Transfer to Object .

If the address owner of the object is a signature-derived address (an account address), then you can use and access it directly as an owned object during the execution of a transaction that address signs. Other addresses cannot access owned objects in any way in a transaction - even to just read the object.

Use address-owned objects in cases where you want or need only a single owner for the object at any point in time. When designing, you should generally prefer address-owned objects over shared objects as address-owned objects whenever it is reasonable or possible to do so. Owned objects do not need to be sequenced through consensus, and are therefore less prone to throughput bottlenecks if the object is being used extensively.

An example of an object that is frequently address-owned is that of a Coin object . If address 0xA11CE had a coin C with 100 SUI and wanted to pay address 0xB0B 100 SUI, 0xA11CE could do so by transferring C to 0xB0B .

This results in C having a new address owner of 0xB0B , and 0xB0B can later use that 100 SUI coin.

Creating address-owned objects

You can use one of the following functions (defined in the transfer module) to create address-owned objects.

You should use the sui::transfer::transfer function if you are defining a custom transfer policy for the object. Use the sui::transfer::public_transfer function to create an address-owned object if the object has the store capability.

After you declare an object as address-owned, its ownership can change over the life of that object - either by adding it as a dynamic object field, transferring it to a different address, or making it immutable. Importantly though, after you create an object and set its ownership, it cannot be shared.

You can access address-owned objects in two different ways, depending on whether or not the address owner of the object corresponds to an object ID.

If the address owner of the object corresponds to an object ID, then you must access and dynamically authenticate it during the execution of the transaction using the mechanisms defined in Transfer to Object .

If the address owner of the object is a signature-derived address (an account address), then you can use and access it directly as an owned object during the execution of a transaction that address signs. Other addresses cannot access owned objects in any way in a transaction - even to just read the object.

Use address-owned objects in cases where you want or need only a single owner for the object at any point in time. When designing, you should generally prefer address-owned objects over shared objects as address-owned objects whenever it is reasonable or possible to do so. Owned objects do not need to be sequenced through consensus, and are therefore less prone to throughput bottlenecks if the object is being used extensively.

An example of an object that is frequently address-owned is that of a Coin object . If address 0xA11CE had a coin C with 100 SUI and wanted to pay address 0xB0B 100 SUI, 0xA11CE could do so by transferring C to 0xB0B .

This results in C having a new address owner of 0xB0B , and 0xB0B can later use that 100 SUI coin.

Accessing address-owned objects

You can access address-owned objects in two different ways, depending on whether or not the address owner of the object corresponds to an object ID.

If the address owner of the object corresponds to an object ID, then you must access and dynamically authenticate it during the execution of the transaction using the mechanisms defined in Transfer to Object .

If the address owner of the object is a signature-derived address (an account address), then you can use and access it directly as an owned object during the execution of a transaction that address signs. Other addresses cannot access owned objects in any way in a transaction - even to just read the object.

Use address-owned objects in cases where you want or need only a single owner for the object at any point in time. When designing, you should generally prefer address-owned objects over shared objects as address-owned objects whenever it is reasonable or possible to do so. Owned objects do not need to be sequenced through consensus, and are therefore less prone to throughput bottlenecks if the object is being used extensively.

An example of an object that is frequently address-owned is that of a Coin object . If address 0xA11CE had a coin C with 100 SUI and wanted to pay address 0xB0B 100 SUI, 0xA11CE could do so by transferring C to 0xB0B .

This results in C having a new address owner of 0xB0B , and 0xB0B can later use that 100 SUI coin.

When to use address-owned objects

Use address-owned objects in cases where you want or need only a single owner for the object at any point in time. When designing, you should generally prefer address-owned objects over shared objects as address-owned objects whenever it is reasonable or possible to do so. Owned objects do not need to be sequenced through consensus, and are therefore less prone to throughput bottlenecks if the object is being used extensively.

An example of an object that is frequently address-owned is that of a Coin object . If address 0xA11CE had a coin C with 100 SUI and wanted to pay address 0xB0B 100 SUI, 0xA11CE could do so by transferring C to 0xB0B .

This results in C having a new address owner of 0xB0B , and 0xB0B can later use that 100 SUI coin.

Example

An example of an object that is frequently address-owned is that of a Coin object . If address 0xA11CE had a coin C with 100 SUI and wanted to pay address 0xB0B 100 SUI, 0xA11CE could do so by transferring C to 0xB0B .

This results in C having a new address owner of 0xB0B , and 0xB0B can later use that 100 SUI coin.