The Move Book

In the Basic Syntax chapter we already covered two out of four abilities - Drop and Copy . They affect the behavior of the value in a scope and are not directly related to storage. It is time to cover the key ability, which allows the struct to be stored.

Historically, the key ability was created to mark the type as a key in the storage . A type with the key ability could be stored at top-level in the storage, and could be directly owned by an account or address. With the introduction of the Object Model , the key ability naturally became the defining ability for the object.

A struct with the key ability is considered an object and can be used in the storage functions. The Sui Verifier will require the first field of the struct to be named id and have the type UID .

A struct with the key ability is still a struct, and can have any number of fields and associated functions. There is no special handling or syntax for packing, accessing or unpacking the struct.

However, because the first field of an object struct must be of type UID - a non-copyable and non-droppable type (we will get to it very soon!), the struct transitively cannot have drop and copy abilities. Thus, the object is non-discardable by design.

Due to the UID requirement for types with key , none of the native types in Move can have the key ability, nor can any of the Standard Library types. The key ability is only present in the Sui Framework and custom types.

Key ability defines the object in Move, and objects are intended to be stored . In the next section we present the sui::transfer module, which provides native storage functions for objects.

Object Definition

A struct with the key ability is considered an object and can be used in the storage functions. The Sui Verifier will require the first field of the struct to be named id and have the type UID .

```bash public struct Object has key { id: UID, // required name: String, }

/// Creates a new Object with a Unique ID public fun new(name: String, ctx: &mut TxContext): Object { Object { id: object::new(ctx), // creates a new UID name, } } ```

A struct with the key ability is still a struct, and can have any number of fields and associated functions. There is no special handling or syntax for packing, accessing or unpacking the struct.

However, because the first field of an object struct must be of type UID - a non-copyable and non-droppable type (we will get to it very soon!), the struct transitively cannot have drop and copy abilities. Thus, the object is non-discardable by design.

Due to the UID requirement for types with key , none of the native types in Move can have the key ability, nor can any of the Standard Library types. The key ability is only present in the Sui Framework and custom types.

Key ability defines the object in Move, and objects are intended to be stored . In the next section we present the sui::transfer module, which provides native storage functions for objects.

## Types with the

Due to the UID requirement for types with key , none of the native types in Move can have the key ability, nor can any of the Standard Library types. The key ability is only present in the Sui Framework and custom types.

Key ability defines the object in Move, and objects are intended to be stored . In the next section we present the sui::transfer module, which provides native storage functions for objects.

## Next Steps

Key ability defines the object in Move, and objects are intended to be stored . In the next section we present the sui::transfer module, which provides native storage functions for objects.

## Further reading
```