The Move Book

Now that you have an understanding of top-level storage functions which are enabled by the key ability, we can talk about the last ability in the list - store .

The store is a special ability that allows a type to be stored in objects. This ability is required for the type to be used as a field in a struct that has the key ability. Another way to put it is that the store ability allows the value to be wrapped in an object.

The store ability also relaxes restrictions on transfer operations. We talk about it more in the Restricted and Public Transfer section.

In previous sections we already used types with the key ability: all objects must have a UID field, which we used in examples; we also used the Storable type as a part of the Config struct. The Config type also has the store ability.

All native types (except for references) in Move have the store ability. This includes:

All of the types defined in the standard library have the store ability as well. This includes:

Definition

The store is a special ability that allows a type to be stored in objects. This ability is required for the type to be used as a field in a struct that has the key ability. Another way to put it is that the store ability allows the value to be wrapped in an object.

The store ability also relaxes restrictions on transfer operations. We talk about it more in the Restricted and Public Transfer section.

In previous sections we already used types with the key ability: all objects must have a UID field, which we used in examples; we also used the Storable type as a part of the Config struct. The Config type also has the store ability.

``bash /// This type has thestore` ability. public struct Storable has store {}

/// Config contains a `Storable` field which must have the `store` ability. public struct Config has key, store { id: UID, stores: Storable, }

/// MegaConfig contains a `Config` field which has the `store` ability. public struct MegaConfig has key { id: UID, config: Config, // there it is! } ```

All native types (except for references) in Move have the store ability. This includes:

All of the types defined in the standard library have the store ability as well. This includes:

Example

In previous sections we already used types with the key ability: all objects must have a UID field, which we used in examples; we also used the Storable type as a part of the Config struct. The Config type also has the store ability.

``bash /// This type has thestore` ability. public struct Storable has store {}

/// Config contains a `Storable` field which must have the `store` ability. public struct Config has key, store { id: UID, stores: Storable, }

/// MegaConfig contains a `Config` field which has the `store` ability. public struct MegaConfig has key { id: UID, config: Config, // there it is! } ```

All native types (except for references) in Move have the store ability. This includes:

All of the types defined in the standard library have the store ability as well. This includes:

Types with the

All native types (except for references) in Move have the store ability. This includes:

All of the types defined in the standard library have the store ability as well. This includes:

Further reading