Owner

An Owner is an entity that can own an object. Each Owner is identified by a SuiAddress which represents either an Address (corresponding to a public key of an account) or an Object, but never both (it is not known up-front whether a given Owner is an Address or an Object).

```
type Owner implements IOwner {
  address: SuiAddress!
  objects(
    first: Int
    after: String
    last: Int
    before: String
    filter: ObjectFilter
  ): MoveObjectConnection!
  balance(
    type: String
  ): Balance
  balances(
    first: Int
    after: String
    last: Int
    before: String
  ): BalanceConnection!
  coins(
    first: Int
    after: String
    last: Int
    before: String
    type: String
  ): CoinConnection!
  stakedSuis(
    first: Int
    after: String
    last: Int
    before: String
  ): StakedSuiConnection!
  defaultSuinsName(
    format: DomainFormat
  ): String
  suinsRegistrations(
    first: Int
    after: String
    last: Int
    before: String
  ): SuinsRegistrationConnection!
  asAddress: Address
  asObject: Object
```

```
  dynamicField(
    name: DynamicFieldName!
  ): DynamicField
  dynamicObjectField(
    name: DynamicFieldName!
  ): DynamicField
  dynamicFields(
    first: Int
    after: String
    last: Int
    before: String
  ): DynamicFieldConnection!
}
```

Fields

`Owner.address.SuiAddress!` non-null scalar

`Owner.objects.MoveObjectConnection!` non-null object

Objects owned by this object or address, optionally `filter`-ed.

`Owner.objects.first.Int` scalar

`Owner.objects.after.String` scalar

`Owner.objects.last.Int` scalar

`Owner.objects.before.String` scalar

`Owner.objects.filter.ObjectFilter` input

`Owner.balance.Balance` object

Total balance of all coins with marker type owned by this object or address. If type is not supplied, it defaults to `0x2::sui::SUI`.

`Owner.balance.type.String` scalar

`Owner.balances.BalanceConnection!` non-null object

The balances of all coin types owned by this object or address.

`Owner.balances.first.Int` scalar

`Owner.balances.after.String` scalar

`Owner.balances.last.Int` scalar

Owner.balances.before.String scalar

Owner.coins.CoinConnection! non-null object

The coin objects for this object or address.

`type` is a filter on the coin's type parameter, defaulting to `0x2::sui::SUI`.

Owner.coins.first.Int scalar

Owner.coins.after.String scalar

Owner.coins.last.Int scalar

Owner.coins.before.String scalar

Owner.coins.type.String scalar

Owner.stakedSuis.StakedSuiConnection! non-null object

The `0x3::staking_pool::StakedSui` objects owned by this object or address.

Owner.stakedSuis.first.Int scalar

Owner.stakedSuis.after.String scalar

Owner.stakedSuis.last.Int scalar

Owner.stakedSuis.before.String scalar

Owner.defaultSuinsName.String scalar

The domain explicitly configured as the default domain pointing to this object or address.

Owner.defaultSuinsName.format.DomainFormat enum

Owner.suinsRegistrations.SuinsRegistrationConnection! non-null object

The SuinsRegistration NFTs owned by this object or address. These grant the owner the capability to manage the associated domain.

Owner.suinsRegistrations.first.Int scalar

Owner.suinsRegistrations.after.String scalar

Owner.suinsRegistrations.last.Int scalar

Owner.suinsRegistrations.before.String scalar

`Owner.asAddress.Address` **object**

`Owner.asObject.Object` **object**

`Owner.dynamicField.DynamicField` **object**

Access a dynamic field on an object using its name. Names are arbitrary Move values whose type have `copy`, `drop`, and `store`, and are specified using their type, and their BCS contents, Base64 encoded.

This field exists as a convenience when accessing a dynamic field on a wrapped object.

`Owner.dynamicField.name.DynamicFieldName!` **non-null** **input**

`Owner.dynamicObjectField.DynamicField` **object**

Access a dynamic object field on an object using its name. Names are arbitrary Move values whose type have `copy`, `drop`, and `store`, and are specified using their type, and their BCS contents, Base64 encoded. The value of a dynamic object field can also be accessed off-chain directly via its address (e.g. using `Query.object`).

This field exists as a convenience when accessing a dynamic field on a wrapped object.

`Owner.dynamicObjectField.name.DynamicFieldName!` **non-null** **input**

`Owner.dynamicFields.DynamicFieldConnection!` **non-null** **object**

The dynamic fields and dynamic object fields on an object.

This field exists as a convenience when accessing a dynamic field on a wrapped object.

`Owner.dynamicFields.first.Int` **scalar**

`Owner.dynamicFields.after.String` **scalar**

`Owner.dynamicFields.last.Int` **scalar**

`Owner.dynamicFields.before.String` **scalar**

Interfaces

`IOwner` **interface**

Interface implemented by GraphQL types representing entities that can own objects. Object owners are identified by an address which can represent either the public key of an account or another object. The same address can only refer to an account or an object, never both, but it is not possible to know which up-front.

Returned By

`owner` query

Member Of

`AddressOwner` object . `BalanceChange` object . `Parent` object . `Validator` object

Edit this page