CoinMetadata

The metadata for a coin type.

```
type CoinMetadata implements IMoveObject, IObject, 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!
  version: UInt53!
  status: ObjectKind!
  digest: String
  owner: ObjectOwner
```

```
  previousTransactionBlock: TransactionBlock
  storageRebate: BigInt
  receivedTransactionBlocks(
    first: Int
    after: String
    last: Int
    before: String
    filter: TransactionBlockFilter
    scanLimit: Int
  ): TransactionBlockConnection!
  bcs: Base64
  contents: MoveValue
  hasPublicTransfer: Boolean!
  display: [DisplayEntry!]
  dynamicField(
    name: DynamicFieldName!
  ): DynamicField
  dynamicObjectField(
    name: DynamicFieldName!
  ): DynamicField
  dynamicFields(
    first: Int
    after: String
    last: Int
    before: String
  ): DynamicFieldConnection!
  decimals: Int
  name: String
  symbol: String
  description: String
  iconUrl: String
  supply: BigInt
}
```

Fields

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

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

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

`CoinMetadata.objects.first.Int` scalar

`CoinMetadata.objects.after.String` scalar

`CoinMetadata.objects.last.Int` scalar

`CoinMetadata.objects.before.String` scalar

CoinMetadata.objects.filter.ObjectFilter input

CoinMetadata.balance.Balance object

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

CoinMetadata.balance.type.String scalar

CoinMetadata.balances.BalanceConnection! non-null object

The balances of all coin types owned by this object.

CoinMetadata.balances.first.Int scalar

CoinMetadata.balances.after.String scalar

CoinMetadata.balances.last.Int scalar

CoinMetadata.balances.before.String scalar

CoinMetadata.coins.CoinConnection! non-null object

The coin objects for this object.

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

CoinMetadata.coins.first.Int scalar

CoinMetadata.coins.after.String scalar

CoinMetadata.coins.last.Int scalar

CoinMetadata.coins.before.String scalar

CoinMetadata.coins.type.String scalar

CoinMetadata.stakedSuis.StakedSuiConnection! non-null object

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

CoinMetadata.stakedSuis.first.Int scalar

CoinMetadata.stakedSuis.after.String scalar

CoinMetadata.stakedSuis.last.Int scalar

CoinMetadata.stakedSuis.before.String scalar

CoinMetadata.defaultSuinsName.String scalar

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

CoinMetadata.defaultSuinsName.format.DomainFormat enum

CoinMetadata.suinsRegistrations.SuinsRegistrationConnection! non-null object

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

CoinMetadata.suinsRegistrations.first.Int scalar

CoinMetadata.suinsRegistrations.after.String scalar

CoinMetadata.suinsRegistrations.last.Int scalar

CoinMetadata.suinsRegistrations.before.String scalar

CoinMetadata.version.UInt53! non-null scalar

CoinMetadata.status.ObjectKind! non-null enum

The current status of the object as read from the off-chain store. The possible states are: NOT_INDEXED, the object is loaded from serialized data, such as the contents of a genesis or system package upgrade transaction. LIVE, the version returned is the most recent for the object, and it is not deleted or wrapped at that version. HISTORICAL, the object was referenced at a specific version or checkpoint, so is fetched from historical tables and may not be the latest version of the object. WRAPPED_OR_DELETED, the object is deleted or wrapped and only partial information can be loaded."

CoinMetadata.digest.String scalar

32-byte hash that identifies the object's contents, encoded as a Base58 string.

CoinMetadata.owner.ObjectOwner union

The owner type of this object: Immutable, Shared, Parent, Address

CoinMetadata.previousTransactionBlock.TransactionBlock object

The transaction block that created this version of the object.

CoinMetadata.storageRebate.BigInt scalar

The amount of SUI we would rebate if this object gets deleted or mutated. This number is recalculated based on the present storage gas price.

CoinMetadata.receivedTransactionBlocks.TransactionBlockConnection! non-null object

The transaction blocks that sent objects to this object.

`scanLimit` restricts the number of candidate transactions scanned when gathering a page of results. It is required for queries that apply more than two complex filters (on function, kind, sender, recipient, input object, changed object, or ids), and can be at most `serviceConfig.maxScanLimit`.

When the scan limit is reached the page will be returned even if it has fewer than `first` results when paginating forward (`last` when paginating backwards). If there are more transactions to scan, `pageInfo.hasNextPage` (or `pageInfo.hasPreviousPage`) will be set to `true`, and `PageInfo.endCursor` (or `PageInfo.startCursor`) will be set to the last transaction that was scanned as opposed to the last (or first) transaction in the page.

Requesting the next (or previous) page after this cursor will resume the search, scanning the next `scanLimit` many transactions in the direction of pagination, and so on until all transactions in the scanning range have been visited.

By default, the scanning range includes all transactions known to GraphQL, but it can be restricted by the `after` and `before` cursors, and the `beforeCheckpoint`, `afterCheckpoint` and `atCheckpoint` filters.

CoinMetadata.receivedTransactionBlocks.first.Int scalar

CoinMetadata.receivedTransactionBlocks.after.String scalar

CoinMetadata.receivedTransactionBlocks.last.Int scalar

CoinMetadata.receivedTransactionBlocks.before.String scalar

CoinMetadata.receivedTransactionBlocks.filter.TransactionBlockFilter input

CoinMetadata.receivedTransactionBlocks.scanLimit.Int scalar

CoinMetadata.bcs.Base64 scalar

The Base64-encoded BCS serialization of the object's content.

CoinMetadata.contents.MoveValue object

Displays the contents of the Move object in a JSON string and through GraphQL types. Also provides the flat representation of the type signature, and the BCS of the corresponding data.

CoinMetadata.**hasPublicTransfer**.Boolean! non-null scalar

Determines whether a transaction can transfer this object, using the TransferObjects transaction command or `sui::transfer::public_transfer`, both of which require the object to have the `key` and `store` abilities.

CoinMetadata.**display**.[DisplayEntry!] list object

The set of named templates defined on-chain for the type of this object, to be handled off-chain. The server substitutes data from the object into these templates to generate a display string per template.

CoinMetadata.**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.

Dynamic fields on wrapped objects can be accessed by using the same API under the Owner type.

CoinMetadata.dynamicField.**name**.**DynamicFieldName!** non-null input

CoinMetadata.**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`).

Dynamic fields on wrapped objects can be accessed by using the same API under the Owner type.

CoinMetadata.dynamicObjectField.**name**.**DynamicFieldName!** non-null input

CoinMetadata.**dynamicFields**.DynamicFieldConnection! non-null object

The dynamic fields and dynamic object fields on an object.

Dynamic fields on wrapped objects can be accessed by using the same API under the Owner type.

CoinMetadata.dynamicFields.**first**.**Int** scalar

CoinMetadata.dynamicFields.**after**.**String** scalar

CoinMetadata.dynamicFields.**last**.**Int** scalar

CoinMetadata.dynamicFields.**before**.**String** scalar

`CoinMetadata.decimals.Int` **scalar**

The number of decimal places used to represent the token.

`CoinMetadata.name.String` **scalar**

Full, official name of the token.

`CoinMetadata.symbol.String` **scalar**

The token's identifying abbreviation.

`CoinMetadata.description.String` **scalar**

Optional description of the token, provided by the creator of the token.

`CoinMetadata.iconUrl.String` **scalar**

`CoinMetadata.supply.BigInt` **scalar**

The overall quantity of tokens that will be issued.

Interfaces

`IMoveObject` **interface**

This interface is implemented by types that represent a Move object on-chain (A Move value whose type has `key`).

`IObject` **interface**

Interface implemented by on-chain values that are addressable by an ID (also referred to as its address). This includes Move objects and packages.

`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

`coinMetadata` **query**

Member Of

MoveObject	object

Edit this page