# CoinMetadata

The metadata for a coin type.

Objects owned by this object, optionally filter -ed.

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

The balances of all coin types owned by this object.

The coin objects for this object.

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

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

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

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

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."

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

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

The transaction block that created this version of the object.

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.

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.

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

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.

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.

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.

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.

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.

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.

The number of decimal places used to represent the token.

Full, official name of the token.

The token's identifying abbreviation.

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

The overall quantity of tokens that will be issued.

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

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

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.

[coinMetadata](#) query

[MoveObject](#) object