[](https://ida.interchain.io/)

[Interchain Developer Academy](https://ida.interchain.io/)/[Interchain Developer Academy](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)



Search

[Interchain Developer Academy](https://ida.interchain.io/)[Interchain Developer Academy](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

Search



Filters

Interchain Developer Academy

[](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Week 0 - Getting Started](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Getting Started](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Blockchain 101](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Blockchain History](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Public and Managed Blockchains](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Consensus in Distributed Networks](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Cryptography](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Self-Assessment Quiz](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Go Introduction - First Steps](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Go Basics](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Go Interfaces](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Control Structures in Go](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Arrays and Slices in Go](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Standard Packages in Go](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Concurrency in Go](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Good-To-Know Dev Terms](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Docker Introduction](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Week 1 - Introduction to the Interchain](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Introduction to the Interchain](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Blockchain Technology and the Interchain](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[The Interchain Ecosystem](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Getting ATOM and Staking It](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[A Blockchain App Architecture](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Accounts](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Transactions](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Messages](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Modules](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Protobuf](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Multistore and Keepers](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[BaseApp](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Queries](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Events](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Context](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Testing](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Relaying with IBC](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Interchain Security](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Bridges](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Migrations](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Week 1 Quiz](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Week 2 - First Steps](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[First Steps](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Setup Your Work Environment](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Run a Node, API, and CLI](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Ignite CLI](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Exercise - Make a Checkers Blockchain](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Store Object](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Create Custom Messages](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Create and Save a Game Properly](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Add a Way to Make a Move](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Emit Game Information](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Record the Game Winner](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Week 2 Exercise](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Week 3 - Introduction to IBC and CosmJS](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Introduction to IBC and CosmJS](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[What is IBC?](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[IBC/TAO - Connections (OPTIONAL)](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[IBC/TAO - Channels (OPTIONAL)](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[IBC/TAO - Clients (OPTIONAL)](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[IBC Token Transfer](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Interchain Accounts (OPTIONAL)](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[IBC Middleware (OPTIONAL)](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Create a Custom IBC Middleware (OPTIONAL)](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Integrate IBC Middleware Into a Chain (OPTIONAL)](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[IBC Tooling](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[What is CosmJS?](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Your First CosmJS Actions](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Compose Complex Transactions](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Learn to Integrate Keplr](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Create Custom CosmJS Interfaces](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Week 4 - Ignite CLI and IBC Advanced](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Ignite CLI and IBC Advanced](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Keep an Up-To-Date Game Deadline](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Keep Track Of How Many Moves Have Been Played](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Put Your Games in Order](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Auto-Expiring Games](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Let Players Set a Wager](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Handle wager payments](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Integration tests](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Incentivize Players](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Help Find a Correct Move](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Play With Cross-Chain Tokens](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Understand IBC Denoms](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Go Relayer](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Hermes Relayer](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Week 5 - CosmJS Advanced](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[CosmJS Advanced](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Create Custom Objects](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Create Custom Messages](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Get an External GUI](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Integrate CosmJS and Keplr](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Backend Script for Game Indexing](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Week 6 - IBC Deep Dive](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[IBC Deep Dive](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[IBC Application Developer Introduction](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Make a Module IBC-Enabled](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Adding Packet and Acknowledgment Data](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Extend the Checkers Game With a Leaderboard](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Create a Leaderboard Chain](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Week 7 - From Code to MVP to Production and Migrations](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[From Code to MVP to Production and Migrations](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Run in Production](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Prepare the Software to Run](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Prepare a Validator and Keys](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Prepare Where the Node Starts](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Prepare and Connect to Other Nodes](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Configure, Run, and Set Up a Service](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Prepare and Do Migrations](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Simulate Production in Docker](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Tally Player Info After Production](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Add a Leaderboard as a Module](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Migrate the Leaderboard Module After Production](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Simulate a Migration in Docker](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Final Exam](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[What's Next?](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

[Continue Your Interchain Journey](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html)

Docs Version Switcher

On this page

[Format](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html#format)

[Parameters](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html#parameters)

[Scope and best practices](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html#scope-and-best-practices)

[Store types](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html#store-types)

[KVStore and Multistore in the Interchain](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html#kvstore-and-multistore-in-the-interchain)

[CacheMultistore](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html#cachemultistore)

[Transient store](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html#transient-store)

[Additional KVStore wrappers](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html#additional-kvstore-wrappers)

[GasKv store](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html#gaskv-store)

[TraceKv store](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html#tracekv-store)

[Prefix store](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html#prefix-store)

[AnteHandler](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html#antehandler)

[Code example](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html#code-example)

[#Copy link](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html#multistore-and-keepers) **Multistore and Keepers**



Look at the following sections before you begin:

* [Transactions](https://ida.interchain.io/academy/2-cosmos-concepts/3-transactions.html)
* [Messages](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)
* [Modules](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)
* [Protobuf](https://ida.interchain.io/academy/2-cosmos-concepts/6-protobuf.html)



Keepers are responsible for managing access to states defined by modules. Because states are accessed through keepers, they are an ideal place to ensure that invariants are enforced and security principles are always applied.   
  
You can find a code example for your checkers blockchain at the end of the section that explores dealing with storage elements, message handling, and gas costs.

A Cosmos SDK application on an application-specific blockchain usually consists of several modules attending to separate concerns. Each module has a state that is a subset of the entire application state.

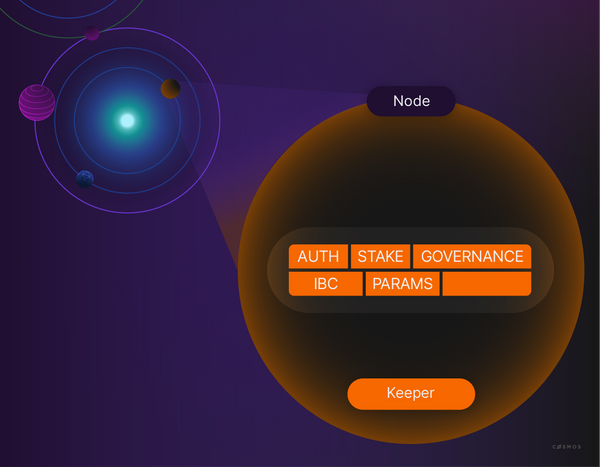
The Cosmos SDK adopts an object-capabilities-based approach to help developers better protect their application from unwanted inter-module interactions. Keepers are at the core of this approach.



A keeper is a Cosmos SDK abstraction that manages access to the subset of the state defined by a module. All access to the module's data must go through the module's keeper.

A keeper can be thought of as the literal gatekeeper of a module's stores. Each store defined within a module (typically an IAVL store) comes with a storeKey which grants unlimited access. The module's keeper holds this storeKey, which should otherwise remain unexposed, and defines methods for reading and writing to any store.

When a module needs to interact with the state defined in another module, it does so by interacting with the methods of the other module’s keeper. Developers control the interactions their module can have with other modules by defining methods and controlling access.



[#Copy link](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html#format) Format

Keepers are generally defined in a /keeper/keeper.go file located in the module’s folder. The type keeper of a module is named simply keeper.go by convention. It usually follows the following structure:



Copy

type Keeper struct {

// Expected external keepers, if any

// Store key(s)

// codec

}

[#Copy link](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html#parameters) Parameters

The following parameters are of importance concerning the type definitions of keepers in modules:

* An expected keeper is a keeper external to a module that is required by the internal keeper of said module. External keepers are listed in the internal keeper's type definition as interfaces. These interfaces are themselves defined in an expected\_keepers.go file in the root of the module's folder. Interfaces are used to reduce the number of dependencies and to facilitate the maintenance of the module in this context.
* storeKeys grant access to any stores of the multistore managed by the module. They should always remain unexposed to external modules.
* cdc is the codec used to marshal and unmarshal structs to and from []byte. The cdc can be codec.BinaryCodec, codec.JSONCodec, or codec.Codec based on your requirements. Note that code.Codec includes the other interfaces. It can be either a proto or amino codec, as long as they implement these interfaces.

Each keeper has its own constructor function, which is called from the application's constructor function. This is where keepers are instantiated and where developers make sure to pass correct instances of the module's keepers to other modules that require them.

[#Copy link](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html#scope-and-best-practices) Scope and best practices

Keepers primarily expose getter and setter methods for stores managed by their module. Methods should remain simple and strictly limited to getting or setting the requested value. Validity checks should already have been done with the ValidateBasic() method of the message and the Msg server before the keeper's methods are called.

The getter method will typically have the following signature:



Copy

func (k Keeper) Get(ctx sdk.Context, key string) (value returnType, found bool)

The setter method will typically have the following signature:



Copy

func (k Keeper) Set(ctx sdk.Context, key string, value valueType)

Keepers also should implement an iterator method with the following signature when appropriate:



Copy

func (k Keeper) GetAll(ctx sdk.Context) (list []returnType)

[#Copy link](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html#store-types) Store types

The Cosmos SDK offers different store types to work with. It is important to gain a good overview of the different store types available for development.

[#Copy link](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html#kvstore-and-multistore-in-the-interchain) KVStore and Multistore in the Interchain

Each Cosmos SDK application contains a state at its root, the Multistore. It is subdivided into separate compartments managed by each module in the application. The Multistore is a store of KVStores that follows the [Multistore interface (opens new window)↗](https://github.com/cosmos/cosmos-sdk/blob/v0.40.0-rc6/store/types/store.go#L104-L133).

The base KVStore and Multistore implementations are wrapped in extensions that offer specialized behavior. A [CommitMultistore (opens new window)↗](https://github.com/cosmos/cosmos-sdk/blob/v0.40.0-rc6/store/types/store.go#L141-L184) is a Multistore with a committer. This is the main type of multistore used in the Cosmos SDK. The underlying KVStore is used primarily to restrict access to the committer.

The [rootMulti.Store (opens new window)↗](https://github.com/cosmos/cosmos-sdk/blob/v0.40.0-rc6/store/rootmulti/store.go#L43-L61) is the go-to implementation of the CommitMultiStore interface. It is a base-layer multistore built around a database on top of which multiple KVStores can be mounted. It is the default multistore store used in BaseApp.

[#Copy link](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html#cachemultistore) CacheMultistore

A [cachemulti.Store (opens new window)↗](https://github.com/cosmos/cosmos-sdk/blob/v0.40.0-rc6/store/cachemulti/store.go#L17-L28) is used whenever the rootMulti.Store needs to be branched. cachemulti.Store branches all substores, creates a virtual store for each substore in its constructor, and holds them in Store.stores. This is used primarily to create an isolated store, typically when it is necessary to mutate the state but it might be reverted later.

CacheMultistore caches all read queries. Store.GetKVStore() returns the store from Store.stores, and Store.Write() recursively calls CacheWrap.Write() on all substores.

[#Copy link](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html#transient-store) Transient store

As the name suggests, Transient.Store is a KVStore that is discarded automatically at the end of each block. Transient.Store is a dbadapter.Store with a dbm.NewMemDB(). All KVStore methods are reused. A new dbadapter.Store is assigned when Store.Commit() is called, discarding the previous reference. Garbage collection is attended to automatically.



Take a closer look at the [IAVL spec (opens new window)↗](https://github.com/cosmos/iavl/blob/v0.15.0-rc5/docs/overview.md) for when working with the IAVL store.



The default implementation of KVStore and CommitKVStore is the IAVL.Store. The IAVL.Store is a self-balancing binary search tree that ensures get and set operations are O(log n), where n is the number of elements in the tree.

[#Copy link](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html#additional-kvstore-wrappers) Additional KVStore wrappers

In addition to the previous store types, there are a few others with more specific usage.

[#Copy link](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html#gaskv-store) GasKv store

Cosmos SDK applications use gas to track resource usage and prevent spam. The GasKv.Store is a KVStore wrapper that enables automatic gas consumption each time a read or write to the store is made. It is the solution of choice to track storage usage in Cosmos SDK applications.

GasKv.Store automatically consumes the appropriate amount of gas depending on the Store.gasConfig when methods of the parent KVStore are called. All KVStores are wrapped in GasKv.Stores by default when retrieved. This is done in the KVStore() method of the context. The default gas configuration is used in this case.

[#Copy link](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html#tracekv-store) TraceKv store

tracekv.Store is a KVStore wrapper which provides operation tracing functionalities over the underlying KVStore. It is applied automatically by the Cosmos SDK on all KVStores if tracing is enabled on the parent MultiStore. When each of the KVStore methods are called, tracekv.Store automatically logs traceOperation to the Store.writer. traceOperation.Metadata is filled with Store.context when it is not nil. TraceContext is a map[string]interface{}.

[#Copy link](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html#prefix-store) Prefix store

prefix.Store is a KVStore wrapper which provides automatic key-prefixing functionalities over the underlying KVStore:

* When Store.{Get, Set}() is called, the store forwards the call to its parent with the key prefixed with the Store.prefix.
* When Store.Iterator() is called, it does not simply prefix the Store.prefix since it does not work as intended. Some of the elements are traversed even when they are not starting with the prefix in this case.

[#Copy link](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html#antehandler) AnteHandler

The AnteHandler is a special handler that implements the AnteHandler interface. It is used to authenticate a transaction before the transaction's internal messages are processed.

The AnteHandler is theoretically optional but still a very important component of public blockchain networks. It serves three primary purposes:

* It is a first line of defense against spam, and the second line of defense (after the mempool) against transaction replay with fees deduction and sequence checking.
* It performs preliminary stateful validity checks, like ensuring signatures are valid, or that a sender has enough funds to pay for fees.
* It plays a role in the incentivization of stakeholders via the collection of transaction fees.

BaseApp holds an AnteHandler as a parameter that is initialized in the application's constructor. The most widely used AnteHandler is the auth module.



For more information on the subject, see the following resources:

* [Cosmos SDK documentation: Gas and Fees (opens new window)↗](https://github.com/cosmos/cosmos-sdk/blob/master/docs/docs/basics/04-gas-fees.md)
* [Cosmos SDK documentation: AnteHandler (opens new window)↗](https://github.com/cosmos/cosmos-sdk/blob/master/docs/docs/basics/04-gas-fees.md#antehandler)

[#Copy link](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html#code-example) Code example

****

**Show me some code for my checkers blockchain**

In the [Accounts section](https://ida.interchain.io/academy/2-cosmos-concepts/2-accounts.html), you were shown the elements of the stored game but not where this game is stored. This will now be explained. **Game object in storage**

You need to decide under what structure you want to store a game in the storage. The Cosmos SDK partitions the global storage per module, with checkers being its own module. You need to take care of how to store games in the checkers module's corner of the key/value pair storage. The first idea is to attribute a unique ID to a game, and to store the game value at that ID. For the sake of clarity, and to differentiate between other stored elements in the future, you add a prefix to each ID. The storage structure looks like this:

Copy

// Pseudo-code

var globalStore sdk.KVStore

checkersStore := globalStore.getAtPrefix("checkers-")

gamesStore := checkersStore.getAtPrefix("games-")

storedGame := gamesStore.getAtPrefix(gameId)

// Or again in a different way

storedGame := globalStore.getAtPrefix("checkers-games-" + gameId)

This is pseudo-code because:

* Prefixes have to be byte[] instead of string. This is easily handled with a cast such as []byte("games-").
* The Cosmos SDK prevents you from directly accessing any random module's store, such that globalStore.getAtPrefix("checkers-") is not allowed and instead has to be accessed via a secret key.

Define the ID of the StoredGame. To return a single object, include Index in the object's value:

Copy

type StoredGame struct {

...

Index string

}

The Cosmos SDK provides much support; you only need to define the required prefixes in your corner of the storage:

Copy

package types

const (

StoredGameKeyPrefix = "StoredGame/value/"

)

This assists you with accessing a game:

Copy

package keeper

import (

"github.com/cosmos/cosmos-sdk/store/prefix"

sdk "github.com/cosmos/cosmos-sdk/store/types"

"github.com/alice/checkers/x/checkers/types"

)

func (k Keeper) GetStoredGame(ctx sdk.Context, gameId string) (storedGame types.StoredGame, found bool) {

checkersStore := ctx.KVStore(k.storeKey)

gamesStore := prefix.NewStore(checkersStore, []byte(types.StoredGameKeyPrefix))

gameBytes := gamesStore.Get([]byte(gameId))

if gameBytes == nil {

return storedGame, false

}

k.cdc.MustUnmarshalBinaryBare(gameBytes, &storedGame)

return storedGame, true

}

If you want to save a game:

Copy

func (k Keeper) SetStoredGame(ctx sdk.Context, storedGame types.StoredGame) {

checkersStore := ctx.KVStore(k.storeKey)

gamesStore := prefix.NewStore(checkersStore, []byte(types.StoredGameKeyPrefix))

gameBytes := k.cdc.MustMarshalBinaryBare(&storedGame)

gamesStore.Set(byte[](storedGame.Index), gameBytes)

}

If you want to delete a stored game, you call gamesStore.Delete(byte[](storedGame.Index)). The KVStore also allows you to obtain an iterator on a given prefix. You can list all stored games because they share the same prefix, which you do with:

Copy

func (k Keeper) GetAllStoredGame(ctx sdk.Context) (list []types.StoredGame) {

checkersStore := ctx.KVStore(k.storeKey)

gamesStore := prefix.NewStore(checkersStore, []byte(types.StoredGameKeyPrefix))

iterator := sdk.KVStorePrefixIterator(gamesStore, []byte{}) // []byte{} is an empty array

defer iterator.Close() // Think of it as: try { everything below } finally { iterator.Close() }

for ; iterator.Valid(); iterator.Next() {

var storedGame types.StoredGame

k.cdc.MustUnmarshalBinaryBare(iterator.Value(), &storedGame)

list = append(list, storedGame)

}

return

}

Note the MustMarshalBinaryBare and MustUnmarshalBinaryBare functions in the previous codec. They need to be instructed on how to proceed with the marshaling. Protobuf handled this for you.

See the [previous section on Protobuf](https://ida.interchain.io/academy/2-cosmos-concepts/6-protobuf.html) to explore how Protobuf deals with the marshaling.

**Boilerplate, boilerplate everywhere!**

Note how the Set, Get, Remove, and GetAll functions shown previously look like boilerplate too. Do you have to redo these functions for every type? *No* - it was all created with this Ignite CLI command:

Copy

$ ignite scaffold map storedGame \

board turn black red wager:uint \

--module checkers \

--no-message

map is the command that tells Ignite CLI to add an Index and store all elements under a map-like structure.

**Other storage elements**

How do you pick a value for the Index in storedGame? A viable idea is to keep a counter in storage for the next game. Unlike StoredGame, which is saved as a map, this SystemInfo object has to be at a unique location in the storage. First define the object:

Copy

package types

type SystemInfo struct {

NextId uint64

}

Then define the key where it resides:

Copy

const (

SystemInfoKey = "SystemInfo-value-"

)

Then define the functions to get and set:

Copy

func (k Keeper) SetSystemInfo(ctx sdk.Context, systemInfo types.SystemInfo) {

systemInfoStore := prefix.NewStore(ctx.KVStore(k.storeKey), []byte(types.SystemInfoKey))

nextBytes := k.cdc.MustMarshalBinaryBare(&systemInfo)

systemInfoStore.Set([]byte{0}, nextBytes)

}

Remember that SystemInfo needs an initial value, which is the role of the genesis block definition:

Copy

type GenesisState struct {

SystemInfo SystemInfo

StoredGameList []StoredGame

}

Now initialize:

Copy

func DefaultGenesis() \*GenesisState {

return &GenesisState{

SystemInfo: SystemInfo{

NextId: uint64(1), // Starts at 1

},

StoredGameList: []StoredGame{}, // Starts empty

}

}

**What about message handling**

You go from the message to the game in storage with MsgCreateGame, which was defined in an earlier [section on messages](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html). That is also the role of the keeper. Define a handling function such as:

Copy

func (k Keeper) CreateGame(goCtx context.Context, msg \*types.MsgCreateGame) (\*types.MsgCreateGameResponse, error) {

ctx := sdk.UnwrapSDKContext(goCtx)

// TODO: Handle the message

return &types.MsgCreateGameResponse{}, nil

}

You now have all the pieces necessary to replace the TODO.

Get the next game ID:

Copy

systemInfo, found := k.GetSystemInfo(ctx)

if !found {

// Panic because this should never happen.

panic("SystemInfo not found")

}

newIndex := strconv.FormatUint(systemInfo.NextId, 10)

Extract and sanitize the message elements:

Copy

creator, err := sdk.AccAddressFromBech32(msg.Creator)

if err != nil {

return nil, errors.New("creator address invalid")

}

black, err := sdk.AccAddressFromBech32(msg.Black)

if err != nil {

// Standard error because users can make mistakes.

return nil, errors.New("black address invalid")

}

red, err := sdk.AccAddressFromBech32(msg.Red)

if err != nil {

return nil, errors.New("red address invalid")

}

Create the stored game object:

Copy

storedGame := types.StoredGame{

Creator: msg.Creator,

Index: newIndex,

Board: rules.New().String(),

Turn: "b",

Black: msg.Black,

Red: msg.Red,

Wager: msg.Wager,

}

Save the stored game object in storage:

Copy

k.SetStoredGame(ctx, storedGame)

Prepare for the next created game:

Copy

systemInfo.NextId++

k.SetSystemInfo(ctx, systemInfo)

Return the game ID for reference:

Copy

return &types.MsgCreateGameResponse{

GameIndex: newIndex,

}, nil

You would also do the same for MsgPlayMoveResponse and MsgRejectGame. Why not try it out as an exercise? See the bottom of the page for relevant links. **More on game theory**

Some players may drop out from games, especially if they know they are about to lose. As the blockchain designer, you need to protect your blockchain, and in particular to avoid bloat, or locked tokens.

A good first point is to introduce a game deadline. This demonstrates how you would add a small feature to your existing blockchain:

Copy

const (

MaxTurnDuration = time.Duration(24 \* 3\_600 \* 1000\_000\_000) // 1 day

DeadlineLayout = "2006-01-02 15:04:05.999999999 +0000 UTC"

)

type StoredGame struct {

...

Deadline string

}

Set its initial value on creation:

Copy

storedGame := types.StoredGame{

...

Deadline: ctx.BlockTime().Add(types.MaxTurnDuration).UTC().Format(types.DeadlineLayout),

}

Update its value after a move:

Copy

storedGame.Deadline = ctx.BlockTime().Add(types.MaxTurnDuration).UTC().Format(types.DeadlineLayout)

Extract and verify its value when necessary:

Copy

deadline, err = time.Parse(DeadlineLayout, storedGame.Deadline)

if err != nil {

panic(err)

}

if deadline.Before(ctx.BlockTime()) {

// TODO

}

**How to expire games**

How can you know what games should be removed? Should you load *all* games and filter for those that have expired? That would be extremely expensive, O(n) of the number of games in fact. This means the more successful your blockchain becomes, the slower it would run.

Better is to use a First-In-First-Out (FIFO) strategy, where fresh games are pushed back to the tail so that the head contains the next games to expire. In the context of the Cosmos SDK, you need to keep track of where the FIFO starts and stops by saving the corresponding game IDs:

Copy

const (

NoFifoIndex = "-1"

)

type SystemInfo struct {

...

FifoHeadIndex string

FifoTailIndex string

}

Each game must know its relative position and the number of moves done, to assist the refunding logic on games with zero, one, or more than two moves:

Copy

type StoredGame struct {

...

MoveCount uint64

BeforeIndex string

AfterIndex string

}

Next, you need to code a regular FIFO, whereby:

* Games are sent to the back when created or played on.
* Games are removed from the FIFO when they are finished or time out.

**When to expire games**

See the next section about the [BaseApp](https://ida.interchain.io/academy/2-cosmos-concepts/8-base-app.html) for a possible solution.



If you want to go beyond out-of-context code samples like the above and see in more detail how to define these features, go to [Run Your Own Cosmos Chain](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/).   
  
More precisely, you can jump to:

* [Store Object - Make a Checkers Blockchain](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html) for general detail of how you handle your game in storage.
* [Create and Save a Game Properly](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/5-create-handling.html) for how you would handle the game when it is being created.
* [Keep an Up-To-Date Game Deadline](https://ida.interchain.io/hands-on-exercise/2-ignite-cli-adv/1-game-deadline.html), where you add a small feature to your chain.
* [Put Your Games in Order](https://ida.interchain.io/hands-on-exercise/2-ignite-cli-adv/3-game-fifo.html) to see the implementation of the FIFO within the constraints of the store.
* [Store leaderboard candidates](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html) in the context's transient store.

synopsis

To summarize, this section has explored:

* How each keeper manages access to the subset of the blockchain state that is a given module's state, which is at the core of the Cosmos SDK's object-capabilities-based approach to protecting applications from unwanted inter-module interactions.
* How each keeper holds a storeKey granting unlimited access to its module's data and defines how to read and write to any store, so when one module needs to interact with another it must follow the methods of the other module's keeper.
* How all Cosmos SDK applications contain a Multistore root state that is subdivided into compartments managed by each module and which stores all the KVStores of the application's modules.
* How inclusion of the AnteHandler component is recommended to authenticate transactions before their internal messages are processed. It defends against spam and other wasteful transaction events, performs preliminary stateful validity checks, and is involved in collecting transaction fees.

previous

[](https://ida.interchain.io/academy/2-cosmos-concepts/6-protobuf.html)

**[Protobuf](https://ida.interchain.io/academy/2-cosmos-concepts/6-protobuf.html)**

up next

**[BaseApp](https://ida.interchain.io/academy/2-cosmos-concepts/8-base-app.html)**

[[](https://ida.interchain.io/academy/2-cosmos-concepts/8-base-app.html)](https://ida.interchain.io/academy/2-cosmos-concepts/8-base-app.html)

Rate this Page

icon smile

icon meh

icon frown

Would you like to add a message?

Submit

Thank you for your Feedback!

[](https://ida.interchain.io/ida-course/discord-info.html)

On this page

[Format](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html#format)

[Parameters](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html#parameters)

[Scope and best practices](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html#scope-and-best-practices)

[Store types](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html#store-types)

[KVStore and Multistore in the Interchain](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html#kvstore-and-multistore-in-the-interchain)

[CacheMultistore](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html#cachemultistore)

[Transient store](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html#transient-store)

[Additional KVStore wrappers](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html#additional-kvstore-wrappers)

[GasKv store](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html#gaskv-store)

[TraceKv store](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html#tracekv-store)

[Prefix store](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html#prefix-store)

[AnteHandler](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html#antehandler)

[Code example](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html#code-example)

#### **Get Cosmos updates**

Unsubscribe at any time. [Privacy Policy↗](https://v1.cosmos.network/privacy)

     Next

Documentation

[Cosmos SDK](https://docs.cosmos.network/)[Cosmos Hub](https://hub.cosmos.network/)[CometBFT](https://docs.cometbft.com/)[IBC Protocol](https://ibc.cosmos.network/)

Community

[Interchain blog](https://blog.cosmos.network/)[Forum](https://forum.cosmos.network/)[Discord](https://discord.gg/cosmosnetwork)

Contributing

[Source code on GitHub](https://github.com/cosmos/sdk-tutorials)

[](https://ida.interchain.io/)

[Interchain Developer Academy](https://ida.interchain.io/)

**[](https://blog.cosmos.network/)[](https://twitter.com/cosmos)[](https://discord.gg/cosmosnetwork)[](https://www.linkedin.com/company/interchain-foundation/about/)[](https://reddit.com/r/cosmosnetwork)[](https://t.me/cosmosproject)[](https://www.youtube.com/c/CosmosProject)**



Dark mode

† This website is maintained by the Interchain Foundation (ICF). The contents and opinions of this website are those of the ICF. The ICF provides links to cryptocurrency exchanges as a service to the public. The ICF does not warrant that the information provided by these websites is correct, complete, and up-to-date. The ICF is not responsible for their content and expressly rejects any liability for damages of any kind resulting from the use, reference to, or reliance on any information contained within these websites.

Cosmos is a registered trademark of the [Interchain Foundation.](https://interchain.io/)[Privacy](https://v1.cosmos.network/privacy)