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

[Interchain Developer Academy](https://ida.interchain.io/)/[Interchain Developer Academy](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)



Search

[Interchain Developer Academy](https://ida.interchain.io/)[Interchain Developer Academy](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

Search



Filters

Interchain Developer Academy

[](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Week 0 - Getting Started](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Getting Started](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Blockchain 101](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Blockchain History](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Public and Managed Blockchains](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Consensus in Distributed Networks](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Cryptography](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Self-Assessment Quiz](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Go Introduction - First Steps](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Go Basics](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Go Interfaces](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Control Structures in Go](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Arrays and Slices in Go](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Standard Packages in Go](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Concurrency in Go](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Good-To-Know Dev Terms](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Docker Introduction](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Week 1 - Introduction to the Interchain](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Introduction to the Interchain](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Blockchain Technology and the Interchain](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[The Interchain Ecosystem](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Getting ATOM and Staking It](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[A Blockchain App Architecture](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Accounts](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Transactions](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Messages](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Modules](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Protobuf](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Multistore and Keepers](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[BaseApp](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Queries](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Events](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Context](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Testing](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Relaying with IBC](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Interchain Security](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Bridges](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Migrations](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Week 1 Quiz](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Week 2 - First Steps](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[First Steps](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Setup Your Work Environment](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Run a Node, API, and CLI](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Ignite CLI](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Exercise - Make a Checkers Blockchain](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Store Object](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Create Custom Messages](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Create and Save a Game Properly](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Add a Way to Make a Move](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Emit Game Information](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Record the Game Winner](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Week 2 Exercise](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Week 3 - Introduction to IBC and CosmJS](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Introduction to IBC and CosmJS](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[What is IBC?](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[IBC/TAO - Connections (OPTIONAL)](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[IBC/TAO - Channels (OPTIONAL)](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[IBC/TAO - Clients (OPTIONAL)](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[IBC Token Transfer](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Interchain Accounts (OPTIONAL)](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[IBC Middleware (OPTIONAL)](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Create a Custom IBC Middleware (OPTIONAL)](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Integrate IBC Middleware Into a Chain (OPTIONAL)](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[IBC Tooling](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[What is CosmJS?](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Your First CosmJS Actions](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Compose Complex Transactions](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Learn to Integrate Keplr](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Create Custom CosmJS Interfaces](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

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

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

[Keep an Up-To-Date Game Deadline](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Keep Track Of How Many Moves Have Been Played](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Put Your Games in Order](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Auto-Expiring Games](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Let Players Set a Wager](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Handle wager payments](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Integration tests](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Incentivize Players](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Help Find a Correct Move](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Play With Cross-Chain Tokens](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Understand IBC Denoms](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Go Relayer](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Hermes Relayer](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Week 5 - CosmJS Advanced](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[CosmJS Advanced](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Create Custom Objects](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Create Custom Messages](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Get an External GUI](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Integrate CosmJS and Keplr](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Backend Script for Game Indexing](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Week 6 - IBC Deep Dive](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[IBC Deep Dive](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[IBC Application Developer Introduction](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Make a Module IBC-Enabled](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Adding Packet and Acknowledgment Data](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Extend the Checkers Game With a Leaderboard](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Create a Leaderboard Chain](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Week 7 - From Code to MVP to Production and Migrations](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[From Code to MVP to Production and Migrations](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Run in Production](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Prepare the Software to Run](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Prepare a Validator and Keys](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Prepare Where the Node Starts](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Prepare and Connect to Other Nodes](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Configure, Run, and Set Up a Service](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Prepare and Do Migrations](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Simulate Production in Docker](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Tally Player Info After Production](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Add a Leaderboard as a Module](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Migrate the Leaderboard Module After Production](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Simulate a Migration in Docker](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Final Exam](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[What's Next?](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

[Continue Your Interchain Journey](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

Docs Version Switcher

On this page

[Messages and the transaction lifecycle](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html#messages-and-the-transaction-lifecycle)

[MsgService](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html#msgservice)

[Client and server code generation](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html#client-and-server-code-generation)

[Code example](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html#code-example)

[#Copy link](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html#messages) **Messages**



It is recommended to take a look at the following previous sections to better understand messages:

* [A Blockchain App Architecture](https://ida.interchain.io/academy/2-cosmos-concepts/1-architecture.html)
* [Accounts](https://ida.interchain.io/academy/2-cosmos-concepts/2-accounts.html)
* [Transactions](https://ida.interchain.io/academy/2-cosmos-concepts/3-transactions.html)



In this section, you will take a closer look at messages, Msg. At the end of the section, you can find a code example that illustrates message creation and the inclusion of messages in transactions for your checkers blockchain.   
  
Understanding Msg will help you prepare for the next section, on [modules in the Cosmos SDK](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html), as messages are a primary object handled by modules.

Messages are one of two primary objects handled by a module in the Cosmos SDK. The other primary object handled by modules is queries. While messages inform the state and have the potential to alter it, queries inspect the module state and are always read-only.

In the Cosmos SDK, a **transaction** contains **one or more messages**. The module processes the messages after the transaction is included in a block by the consensus layer.

****

**Signing a message**

Remember from the [previous section on transactions](https://ida.interchain.io/academy/2-cosmos-concepts/3-transactions.html) that transactions must be signed before a validator includes them in a block. Every message in a transaction must be signed by the addresses as specified by GetSigners. The Cosmos SDK currently allows signing transactions with either SIGN\_MODE\_DIRECT or SIGN\_MODE\_LEGACY\_AMINO\_JSON methods. When an account signs a message it signs an array of bytes. This array of bytes is the outcome of serializing the message. For the signature to be verifiable at a later date, this conversion needs to be deterministic. For this reason, you define a canonical bytes-representation of the message, typically with the parameters ordered alphabetically.

[#Copy link](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html#messages-and-the-transaction-lifecycle) Messages and the transaction lifecycle

Transactions containing one or more valid messages are serialized and confirmed by CometBFT. As you might recall, CometBFT is agnostic to the transaction interpretation and has absolute finality. When a transaction is included in a block, it is confirmed and finalized with no possibility of chain re-organization or cancellation.

The confirmed transaction is relayed to the Cosmos SDK application for interpretation. Each message is routed to the appropriate module via BaseApp using MsgServiceRouter. BaseApp decodes each message contained in the transaction. Each module has its own MsgService that processes each received message.

[#Copy link](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html#msgservice) MsgService

Although it is technically feasible to proceed to create a novel MsgService, the recommended approach is to define a Protobuf Msg service. Each module has exactly one Protobuf Msg service defined in tx.proto and there is an RPC service method for each message type in the module. The Protobuf message service implicitly defines the interface layer of the state, mutating processes contained within the module.

How does all of this translate into code? Here is an example MsgService from the [bank module (opens new window)↗](https://docs.cosmos.network/v0.46/modules/bank/):



Copy

// Msg defines the bank Msg service.

service Msg {

// Send defines a method for sending coins from one account to another account.

rpc Send(MsgSend) returns (MsgSendResponse);

// MultiSend defines a method for sending coins from some accounts to other accounts.

rpc MultiSend(MsgMultiSend) returns (MsgMultiSendResponse);

}

In this example:

* Each Msg service method has exactly **one argument**, such as MsgSend, which must implement the sdk.Msg interface and a Protobuf response.
* The **standard naming convention** is to call the RPC argument Msg<service-rpc-name> and the RPC response Msg<service-rpc-name>Response.

[#Copy link](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html#client-and-server-code-generation) Client and server code generation

The Cosmos SDK uses Protobuf definitions to generate client and server code:

* The MsgServer interface defines the server API for the Msg service. Its implementation is described in the [Msg services documentation (opens new window)↗](https://docs.cosmos.network/main/building-modules/msg-services.html).
* Structures are generated for all RPC requests and response types.



If you want to dive deeper when it comes to messages, the Msg service, and modules, see:

* The Cosmos SDK documentation on [Msg service (opens new window)↗](https://docs.cosmos.network/main/building-modules/msg-services.html).
* The Cosmos SDK documentation on messages and queries, addressing how to define messages using Msg services - [Amino LegacyMsg (opens new window)↗](https://docs.cosmos.network/main/building-modules/messages-and-queries.html#legacy-amino-legacymsgs).

[#Copy link](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html#code-example) Code example

****

**Show me some code for my checkers blockchain - Including messages**

In the [previous](https://ida.interchain.io/academy/2-cosmos-concepts/3-transactions.html) design exercise's code examples, the ABCI application was aware of a single transaction type: that of a checkers move with four int values. With multiple games, this is no longer sufficient. Additionally, you need to conform to the SDK's way of handling Tx, which means **creating messages that are then included in a transaction**.

If you want the guided coding exercise instead of design and implementation considerations, see the links at the bottom of the page.

**What you need**

Begin by describing the messages you need for your checkers application to have a solid starting point before diving into the code:

1. In the former *Play* transaction, your four integers need to move from the transaction to an sdk.Msg, wrapped in said transaction. Four flat int values are no longer sufficient, as you need to follow the sdk.Msg interface, identify the game for which a move is meant, and distinguish a move message from other message types.
2. You need to add a message type for creating a new game. When this is done, a player can create a new game which mentions other players. A generated (possibly) ID identifies this newly created game and is returned to the message creator.
3. It would be a good feature for the other person to be able to reject the challenge. This would have the added benefit of clearing the state of stale, unstarted games.

**How to proceed**

Focus on the messages around the **game creation**. There is no single true way of deciding what goes into your messages. The following is one reasonable example.

1. The message itself is structured like this:

Copy

type MsgCreateGame struct {

Creator string

Black string

Red string

}

Note that Creator contains the address of the message signer.

1. The corresponding response message would then be:

Copy

type MsgCreateGameResponse struct {

GameIndex string

}

The idea here is that when the creator does not know the ID of the game that will be created, the ID needs to be returned.

With the messages defined, you need to declare how the message should be handled. This involves:

1. Describing how the messages are serialized.
2. Writing the code that handles the message and places the new game in the storage.
3. Putting hooks and callbacks at the right places in the general message handling.

Thinking from design to implementation, Ignite CLI can help you create these elements, plus the MsgCreateGame and MsgCreateGameResponse objects, with this command:

Copy

$ ignite scaffold message createGame black red \

--module checkers \

--response gameIndex

Ignite CLI creates a variety of other files. See [Run Your Own Cosmos Chain](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/) for details, and to make additions to existing files.

***A sample of things Ignite CLI did for you***

Ignite CLI significantly reduces the amount of work a developer has to do to build an application with the Cosmos SDK. Among others, it assists with:

1. Getting the signer, the Creator, of your message:

Copy

func (msg \*MsgCreateGame) GetSigners() []sdk.AccAddress {

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

if err != nil {

panic(err)

}

return []sdk.AccAddress{creator}

}

Where GetSigners is [a requirement of sdk.Msg (opens new window)](https://github.com/cosmos/cosmos-sdk/blob/1dba673/types/tx_msg.go#L21).

1. Making sure the message's bytes to sign are deterministic:

Copy

func (msg \*MsgCreateGame) GetSignBytes() []byte {

bz := ModuleCdc.MustMarshalJSON(msg)

return sdk.MustSortJSON(bz)

}

1. Adding a callback for your new message type in your module's message handler x/checkers/handler.go:

Copy

...

switch msg := msg.(type) {

...

case \*types.MsgCreateGame:

res, err := msgServer.CreateGame(sdk.WrapSDKContext(ctx), msg)

return sdk.WrapServiceResult(ctx, res, err)

...

}

1. Creating an empty shell of a file (x/checkers/keeper/msg\_server\_create\_game.go) for you to include your code, and the response message:

Copy

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

ctx := sdk.UnwrapSDKContext(goCtx)

// TODO: Handling the message

\_ = ctx

return &types.MsgCreateGameResponse{}, nil

}

Ignite CLI is opinionated in terms of which files it creates to separate which concerns. If you are not using it, you are free to create the files you want.

**What is left to do?**

Your work is mostly done. You want to create the specific game creation code to replace // TODO: Handling the message. For this, you need to:

1. Decide how to create a new and unique game ID: newIndex.

For more details, and to avoid diving too deep in this section, see:

* + [Run Your Own Cosmos Chain](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/) to start the guided coding exercise from scratch,
  + [Create and Save a Game Properly](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/5-create-handling.html) to jump straight where you handle the new message.

1. Extract and verify addresses, such as:

Copy

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

if err != nil {

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

}

1. Create a game object with all required parameters - see the [modules section](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html) for the declaration of this object:

Copy

storedGame := {

Creator: creator,

Index: newIndex,

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

Black: black,

Red: red,

}

1. Send the game object to storage - see the [modules section](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html) for the declaration of this function:

Copy

k.Keeper.SetStoredGame(ctx, storedGame)

1. And finally, return the expected message:

Copy

return &types.MsgCreateGameResponse{

GameIndex: newIndex,

}, nil

Remember, as a part of good design practice:

* If you encounter an internal error (one that denotes an error in logic or catastrophic failure), you should panic("This situation should not happen").
* If you encounter a user or *regular* error (like a user not having enough funds), you should return a regular error.

**The other messages**

You can also implement other messages:

1. The **play message**, which means implicitly accepting the challenge when playing for the first time. If you create it with Ignite CLI, use:

Copy

$ ignite scaffold message playMove gameIndex fromX:uint fromY:uint toX:uint toY:uint --module checkers --response capturedX:int,capturedY:int,winner

This generates, among others, the object files, callbacks, and a new file for you to write your code:

Copy

func (k msgServer) PlayMove(goCtx context.Context, msg \*types.MsgPlayMove) (\*types.MsgPlayMoveResponse, error) {

ctx := sdk.UnwrapSDKContext(goCtx)

// TODO: Handling the message

\_ = ctx

return &types.MsgPlayMoveResponse{}, nil

}

To jump straight to the corresponding part of the coding exercise, head to [Add a Way to Make a Move](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/6-play-game.html).

1. The **reject message**, which should be valid only if the player never played any moves in this game.

Copy

$ ignite scaffold message rejectGame gameIndex --module checkers

This generates, among others:

Copy

func (k msgServer) RejectGame(goCtx context.Context, msg \*types.MsgRejectGame) (\*types.MsgRejectGameResponse, error) {

ctx := sdk.UnwrapSDKContext(goCtx)

// TODO: Handling the message

\_ = ctx

return &types.MsgRejectGameResponse{}, nil

}

This message is not implemented in the coding exercise as it is not necessary if you implement an expiration.

**Other considerations**

What would happen if one of the two players has accepted the game by playing, but the other player has neither accepted nor rejected the game? You can address this scenario by:

* Having a timeout after which the game is canceled.
* Keeping an index as a First-In-First-Out (FIFO) list, or a list of unstarted games ordered by their cancellation time, so that this automatic trigger does not consume too many resources.

What would happen if a player stops taking turns? To ensure functionality for your checkers application, you can consider:

* Having a timeout after which the game is forfeited. You could also automatically charge the forgetful player, if and when you implement a wager system. For the guided coding exercise on this part, head straight to [Keep an Up-To-Date Game Deadline](https://ida.interchain.io/hands-on-exercise/2-ignite-cli-adv/1-game-deadline.html).
* Keeping an index of games that could be forfeited. If both timeouts are the same, you can keep a single FIFO list of games so you can clear them from the top of the list as necessary. For the guided coding exercise on this part, head straight to [Put Your Games in Order](https://ida.interchain.io/hands-on-exercise/2-ignite-cli-adv/3-game-fifo.html).
* Handling the cancelation in ABCI's EndBlock (or rather its equivalent in the Cosmos SDK) without any of the players having to trigger the cancelation. For the guided coding exercise on this part, head straight to [Auto-Expiring Games](https://ida.interchain.io/hands-on-exercise/2-ignite-cli-adv/4-game-forfeit.html).

In general terms, you could add timeout: Timestamp to your StoredGame and update it every time something changes in the game. You can decide on a maximum delay, for example *one day*.

There are no *open* challenges, meaning a player cannot create a game where the second player is unknown until someone steps in. Therefore, player matching is left outside of the blockchain. The enterprising student can incorporate it inside the blockchain by changing the necessary models.



If you would like to get started on building your own checkers game, you can go straight to the main exercise in [Run Your Own Cosmos Chain](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/) to start from scratch.

More specifically, you can jump to:

* [Create Custom Messages](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/4-create-message.html) to see how to simply create the MsgCreateGame,
* [Create and Save a Game Properly](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/5-create-handling.html) to see how to handle MsgCreateGame,
* [Add a Way to Make a Move](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/6-play-game.html) for the same but with MsgPlayMove.

synopsis

To summarize, this section has explored:

* Messages, one of two primary objects handled by a module in the Cosmos SDK, which inform the state and have the potential to alter it.
* How one or more messages form a transaction in the Cosmos SDK, and messages are only processed after a transaction is signed by a validator and included in a block by the consensus layer.
* An example of more complex message handling capabilities related to the checkers game blockchain.

previous

[](https://ida.interchain.io/academy/2-cosmos-concepts/3-transactions.html)

**[Transactions](https://ida.interchain.io/academy/2-cosmos-concepts/3-transactions.html)**

up next

**[Modules](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)**

[[](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.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

[Messages and the transaction lifecycle](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html#messages-and-the-transaction-lifecycle)

[MsgService](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html#msgservice)

[Client and server code generation](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html#client-and-server-code-generation)

[Code example](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.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)