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

[Interchain Developer Academy](https://ida.interchain.io/)/[Interchain Developer Academy](https://ida.interchain.io/academy/3-ibc/2-connections.html)



Search

[Interchain Developer Academy](https://ida.interchain.io/)[Interchain Developer Academy](https://ida.interchain.io/academy/3-ibc/2-connections.html)

Search



Filters

Interchain Developer Academy

[](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Week 0 - Getting Started](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Getting Started](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Blockchain 101](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Blockchain History](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Public and Managed Blockchains](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Consensus in Distributed Networks](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Cryptography](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Self-Assessment Quiz](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Go Introduction - First Steps](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Go Basics](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Go Interfaces](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Control Structures in Go](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Arrays and Slices in Go](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Standard Packages in Go](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Concurrency in Go](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Good-To-Know Dev Terms](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Docker Introduction](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Week 1 - Introduction to the Interchain](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Introduction to the Interchain](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Blockchain Technology and the Interchain](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[The Interchain Ecosystem](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Getting ATOM and Staking It](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[A Blockchain App Architecture](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Accounts](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Transactions](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Messages](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Modules](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Protobuf](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Multistore and Keepers](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[BaseApp](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Queries](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Events](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Context](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Testing](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Relaying with IBC](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Interchain Security](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Bridges](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Migrations](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Week 1 Quiz](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Week 2 - First Steps](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[First Steps](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Setup Your Work Environment](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Run a Node, API, and CLI](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Ignite CLI](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Exercise - Make a Checkers Blockchain](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Store Object](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Create Custom Messages](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Create and Save a Game Properly](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Add a Way to Make a Move](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Emit Game Information](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Record the Game Winner](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Week 2 Exercise](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Week 3 - Introduction to IBC and CosmJS](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Introduction to IBC and CosmJS](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[What is IBC?](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[IBC/TAO - Connections (OPTIONAL)](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[IBC/TAO - Channels (OPTIONAL)](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[IBC/TAO - Clients (OPTIONAL)](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[IBC Token Transfer](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Interchain Accounts (OPTIONAL)](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[IBC Middleware (OPTIONAL)](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Create a Custom IBC Middleware (OPTIONAL)](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Integrate IBC Middleware Into a Chain (OPTIONAL)](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[IBC Tooling](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[What is CosmJS?](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Your First CosmJS Actions](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Compose Complex Transactions](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Learn to Integrate Keplr](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Create Custom CosmJS Interfaces](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Week 4 - Ignite CLI and IBC Advanced](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Ignite CLI and IBC Advanced](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Keep an Up-To-Date Game Deadline](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Keep Track Of How Many Moves Have Been Played](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Put Your Games in Order](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Auto-Expiring Games](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Let Players Set a Wager](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Handle wager payments](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Integration tests](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Incentivize Players](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Help Find a Correct Move](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Play With Cross-Chain Tokens](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Understand IBC Denoms](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Go Relayer](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Hermes Relayer](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Week 5 - CosmJS Advanced](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[CosmJS Advanced](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Create Custom Objects](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Create Custom Messages](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Get an External GUI](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Integrate CosmJS and Keplr](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Backend Script for Game Indexing](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Week 6 - IBC Deep Dive](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[IBC Deep Dive](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[IBC Application Developer Introduction](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Make a Module IBC-Enabled](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Adding Packet and Acknowledgment Data](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Extend the Checkers Game With a Leaderboard](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Create a Leaderboard Chain](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Week 7 - From Code to MVP to Production and Migrations](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[From Code to MVP to Production and Migrations](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Run in Production](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Prepare the Software to Run](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Prepare a Validator and Keys](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Prepare Where the Node Starts](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Prepare and Connect to Other Nodes](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Configure, Run, and Set Up a Service](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Prepare and Do Migrations](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Simulate Production in Docker](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Tally Player Info After Production](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Add a Leaderboard as a Module](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Migrate the Leaderboard Module After Production](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Simulate a Migration in Docker](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Final Exam](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[What's Next?](https://ida.interchain.io/academy/3-ibc/2-connections.html)

[Continue Your Interchain Journey](https://ida.interchain.io/academy/3-ibc/2-connections.html)

Docs Version Switcher

On this page

[Connections](https://ida.interchain.io/academy/3-ibc/2-connections.html#connections)

[Version negotiation](https://ida.interchain.io/academy/3-ibc/2-connections.html#version-negotiation)

[Connection handshakes and states](https://ida.interchain.io/academy/3-ibc/2-connections.html#connection-handshakes-and-states)

[Handshake step 1 - ConnOpenInit](https://ida.interchain.io/academy/3-ibc/2-connections.html#handshake-step-1-connopeninit)

[Handshake step 2 - ConnOpenTry](https://ida.interchain.io/academy/3-ibc/2-connections.html#handshake-step-2-connopentry)

[Handshake step 3 - ConnOpenAck](https://ida.interchain.io/academy/3-ibc/2-connections.html#handshake-step-3-connopenack)

[Handshake step 4 - ConnOpenConfirm](https://ida.interchain.io/academy/3-ibc/2-connections.html#handshake-step-4-connopenconfirm)

[Crossing hellos](https://ida.interchain.io/academy/3-ibc/2-connections.html#crossing-hellos)

[An imposter](https://ida.interchain.io/academy/3-ibc/2-connections.html#an-imposter)

[#Copy link](https://ida.interchain.io/academy/3-ibc/2-connections.html#transport-authentication-and-ordering-layer-connections) **Transport, Authentication, and Ordering Layer - Connections**



IBC in depth. Discover the IBC protocol in detail:

* Learn more about connection negotiation.
* Explore connection states.
* How IBC repels hostile connection attempts.

Now that you covered the introduction and have a better understanding of how different Inter-Blockchain Communication Protocol (IBC) components and Interchain Standards (ICS) relate to each other, take a deep dive into IBC/TAO (transport, authentication, and ordering) and the [IBC module (opens new window)↗](https://github.com/cosmos/ibc-go).

[#Copy link](https://ida.interchain.io/academy/3-ibc/2-connections.html#connections) Connections

If you want to connect two blockchains with IBC, you will need to establish an IBC **connection**. Connections, established by a four-way handshake, are responsible for:

1. Establishing the identity of the counterparty chain.
2. Preventing a malicious entity from forging incorrect information by pretending to be the counterparty chain. IBC connections are established by on-chain ledger code and therefore do not require interaction with off-chain (trusted) third-party processes.



The connection semantics are described in [ICS-3 (opens new window)↗](https://github.com/cosmos/ibc/tree/master/spec/core/ics-003-connection-semantics).

In the IBC stack, connections are built on top of clients, so technically there could be multiple connections for each client if the client is interacting with multiple versions of the IBC protocol. For now, the setup should connote one connection for each client.



[#Copy link](https://ida.interchain.io/academy/3-ibc/2-connections.html#version-negotiation) Version negotiation

Note that versioning here refers to the IBC protocol spec and not the ibc-go module. A backwards incompatible update is currently not planned.



Copy

// Version defines the versioning scheme used to negotiate the IBC verison in

// the connection handshake.

type Version struct {

// unique version identifier

Identifier string `protobuf:"bytes,1,opt,name=identifier,proto3" json:"identifier,omitempty"`

// list of features compatible with the specified identifier

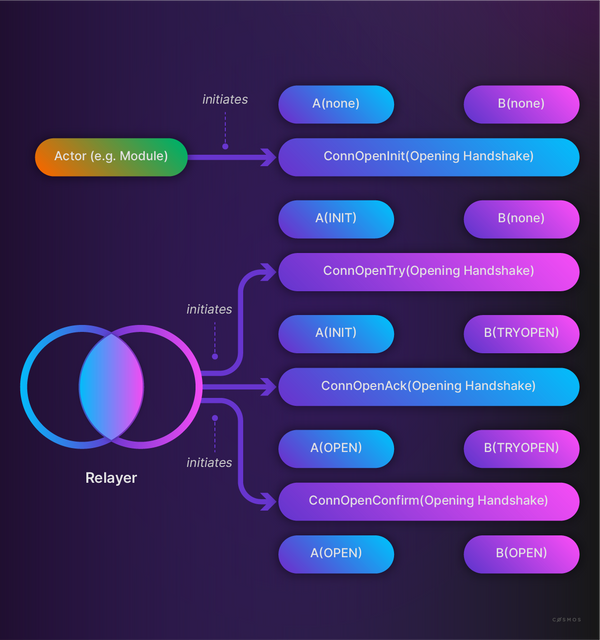
Features []string `protobuf:"bytes,2,rep,name=features,proto3" json:"features,omitempty"`

}

Protocol versioning is important to establish, as different protocol versions may not be compatible, for example due to proofs being stored on a different path. There are three types of protocol version negotiation:

1. *Default, no selection*: only one protocol version is supported. This is default to propose.
2. *With selection*: two protocol versions can be proposed, such that the chain initiating OpenInit or OpenTry has a choice of which version to go with.
3. *Impossible communication*: a backwards incompatible IBC protocol version. For example, if an IBC module changes where it stores its proofs (proof paths), errors result. There are no plans to upgrade to a backwards incompatible IBC protocol version.

As discussed previously, the opening handshake protocol allows each chain to verify the identifier used to reference the connection on the other chain, enabling modules on each chain to reason about the reference of the other chain.



With regards to the connection on the other side, the [connection protobufs (opens new window)↗](https://github.com/cosmos/ibc-go/blob/v5.1.0/proto/ibc/core/connection/v1/connection.proto) contains the Counterparty definition:



Copy

// Counterparty defines the counterparty chain associated with a connection end.

message Counterparty {

option (gogoproto.goproto\_getters) = false;

// identifies the client on the counterparty chain associated with a given

// connection.

string client\_id = 1 [(gogoproto.moretags) = "yaml:\"client\_id\""];

// identifies the connection end on the counterparty chain associated with a

// given connection.

string connection\_id = 2 [(gogoproto.moretags) = "yaml:\"connection\_id\""];

// commitment merkle prefix of the counterparty chain.

ibc.core.commitment.v1.MerklePrefix prefix = 3 [(gogoproto.nullable) = false];

}

In this definition, connection-id is used as a key to map and retrieve connections associated with a certain client from the store.

prefix is used by the clients to construct merkle prefix paths which are then used to verify proofs.

[#Copy link](https://ida.interchain.io/academy/3-ibc/2-connections.html#connection-handshakes-and-states) Connection handshakes and states

Establishing an IBC connection (for example, between chain A and chain B) requires four handshakes:

1. OpenInit
2. OpenTry
3. OpenAck
4. OpenConfirm

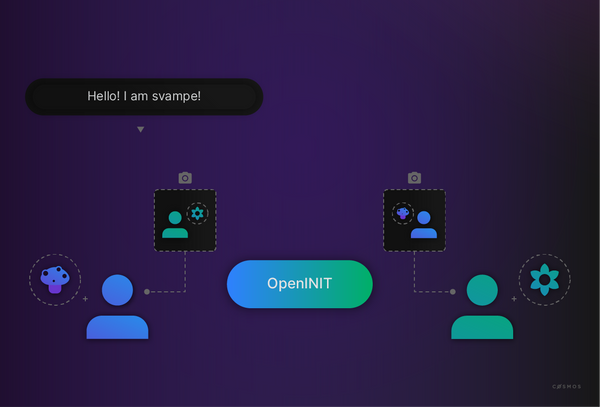


Colin Axnér of Interchain gives an overview of how IBC Connections work (ICS-03), along with a code walkthrough, in the context of the Inter-Blockchain Communications Protocol (IBC).

A high level overview of a successful four-way handshake is as follows:

[#Copy link](https://ida.interchain.io/academy/3-ibc/2-connections.html#handshake-step-1-connopeninit) Handshake step 1 - ConnOpenInit

OpenInit initializes any connection which may occur, while still necessitating agreement from both sides. It is like an identifying announcement from the IBC module on chain A which is submitted by a relayer. The relayer should also submit a MsgUpdateClient with chain A as the source chain before this handshake. MsgUpdateClient updates the client on the initializing chain A with the latest consensus state of chain B.



The initiation of this handshake from chain A updates its connection state to INIT.

OpenInit proposes a protocol version to be used for the IBC connection. A relayer-submitted OpenInit which contains a protocol version that is not supported by chain A will be expected to fail.

The reference implementation for the connection handshake is found in the [IBC module repository (opens new window)↗](https://github.com/cosmos/ibc-go/blob/v5.1.0/modules/core/03-connection/keeper/handshake.go). Examine ConnOpenInit:



Copy

func (k Keeper) ConnOpenInit(

ctx sdk.Context,

clientID string,

counterparty types.Counterparty, // counterpartyPrefix, counterpartyClientIdentifier

version \*types.Version,

delayPeriod uint64,

) (string, error) {

... //version negotiation logic

// connection defines chain A's ConnectionEnd

connectionID := k.GenerateConnectionIdentifier(ctx)

connection := types.NewConnectionEnd(types.INIT, clientID, counterparty, types.ExportedVersionsToProto(versions), delayPeriod)

k.SetConnection(ctx, connectionID, connection)

if err := k.addConnectionToClient(ctx, clientID, connectionID); err != nil {

return "", err

}

k.Logger(ctx).Info("connection state updated", "connection-id", connectionID, "previous-state", "NONE", "new-state", "INIT")

defer func() {

telemetry.IncrCounter(1, "ibc", "connection", "open-init")

}()

EmitConnectionOpenInitEvent(ctx, connectionID, clientID, counterparty)

return connectionID, nil

}

This function creates a unique connectionID. It adds the connection to a list of connections associated with a specific client.

It creates a new ConnectionEnd:



Copy

//@ func (k Keeper) ConnOpenInit

...

connection := types.NewConnectionEnd(types.INIT, clientID, counterparty, types.ExportedVersionsToProto(versions), delayPeriod)

k.SetConnection(ctx, connectionID, connection)

...

With the following proto definition:



Copy

// ConnectionEnd defines a stateful object on a chain connected to another separate one.

// NOTE: there must only be 2 defined ConnectionEnds to establish

// a connection between two chains, so the connections are mapped and stored as `ConnectionEnd` on the respective chains.

message ConnectionEnd {

option (gogoproto.goproto\_getters) = false;

// client associated with this connection.

string client\_id = 1 [(gogoproto.moretags) = "yaml:\"client\_id\""];

// IBC version which can be utilised to determine encodings or protocols for

// channels or packets utilising this connection.

repeated Version versions = 2;

// current state of the connection end.

State state = 3;

// counterparty chain associated with this connection.

Counterparty counterparty = 4 [(gogoproto.nullable) = false];

// delay period that must pass before a consensus state can be used for

// packet-verification NOTE: delay period logic is only implemented by some

// clients.

uint64 delay\_period = 5 [(gogoproto.moretags) = "yaml:\"delay\_period\""];

}

ConnOpenInit is triggered by the **relayer**, which constructs the message and sends it to the SDK that uses the [msg\_server.go (opens new window)↗](https://github.com/cosmos/ibc-go/blob/v5.1.0/modules/core/keeper/msg_server.go) previously seen to call ConnOpenInit:



Copy

// ConnectionOpenInit defines a rpc handler method for MsgConnectionOpenInit.

func (k Keeper) ConnectionOpenInit(goCtx context.Context, msg \*connectiontypes.MsgConnectionOpenInit) (\*connectiontypes.MsgConnectionOpenInitResponse, error) {

ctx := sdk.UnwrapSDKContext(goCtx)

if \_, err := k.ConnectionKeeper.ConnOpenInit(ctx, msg.ClientId, msg.Counterparty, msg.Version, msg.DelayPeriod); err != nil {

return nil, sdkerrors.Wrap(err, "connection handshake open init failed")

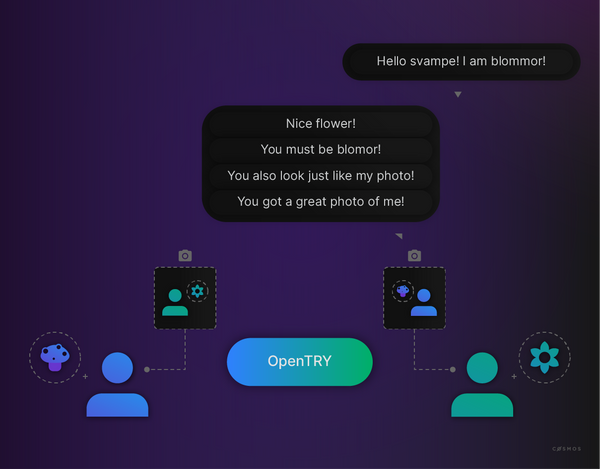
}

return &connectiontypes.MsgConnectionOpenInitResponse{}, nil

}

[#Copy link](https://ida.interchain.io/academy/3-ibc/2-connections.html#handshake-step-2-connopentry) Handshake step 2 - ConnOpenTry

OpenInit is followed by an OpenTry response, in which chain B verifies the identity of chain A according to information that chain B has about chain A in its light client (the algorithm and the last snapshot of the consensus state containing the root hash of the latest height as well as the next validator set). It also responds to some of the information about its own identity in the OpenInit announcement from chain A.



The purpose of this step of the handshake is double verification: not only for chain B to verify that chain A is the expected counterparty identity, but also to verify that the counterparty has accurate information about chain B's identity. The relayer also submits two MsgUpdateClients with chain A and chain B as source chains before this handshake. These update the light clients of both chain A and chain B in order to make sure that the state verifications in this step are successful.

The initiation of this handshake from chain B updates its connection state to TRYOPEN.

With regards to IBC protocol versioning, OpenTry either accepts the protocol version which has been proposed in OpenInit or proposes another protocol version from the versions available to chain A to be used for the IBC connection. A relayer-submitted OpenTry which contains an unsupported protocol version will be expected to fail.

The [implementation of OpenTry (opens new window)↗](https://github.com/cosmos/ibc-go/blob/v5.1.0/modules/core/03-connection/keeper/handshake.go#L61-L147) is as follows:



Copy

// ConnOpenTry relays notice of a connection attempt on chain A to chain B (this

// code is executed on chain B).

//

// NOTE:

// - Here chain A acts as the counterparty

// - Identifiers are checked on msg validation

func (k Keeper) ConnOpenTry(

ctx sdk.Context,

counterparty types.Counterparty, // counterpartyConnectionIdentifier, counterpartyPrefix and counterpartyClientIdentifier

delayPeriod uint64,

clientID string, // clientID of chainA

clientState exported.ClientState, // clientState that chainA has for chainB

counterpartyVersions []exported.Version, // supported versions of chain A

proofInit []byte, // proof that chainA stored connectionEnd in state (on ConnOpenInit)

proofClient []byte, // proof that chainA stored a light client of chainB

proofConsensus []byte, // proof that chainA stored chainB's consensus state at consensus height

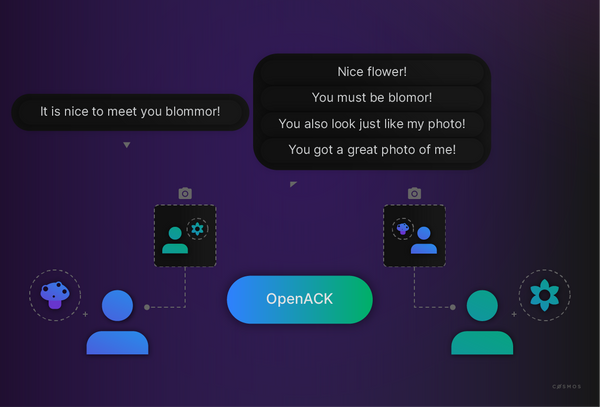
proofHeight exported.Height, // height at which relayer constructs proof of A storing connectionEnd in state

consensusHeight exported.Height, // latest height of chain B which chain A has stored in its chain B client

) ...

[#Copy link](https://ida.interchain.io/academy/3-ibc/2-connections.html#handshake-step-3-connopenack) Handshake step 3 - ConnOpenAck

OpenAck is very similar to the functionality of OpenInit, except that the information verification now occurs for chain A. As in OpenTry, the relayer also submits two MsgUpdateClients with chain A and chain B as source chains before this handshake. These update the light clients of both chain A and chain B, in order to make sure that the state verifications in this step are successful.



The initiation of this handshake from chain A updates its connection state to OPEN. It is important to note that the counterparty chain *must* have a TRYOPEN connection state in order for the handshake and connection state update to be successful.

With regards to version negotiation, OpenAck must confirm the protocol version which has been proposed in OpenTry. It ends the connection handshake process if the version is unwanted or unsupported.

The [OpenAck code (opens new window)↗](https://github.com/cosmos/ibc-go/blob/v5.1.0/modules/core/03-connection/keeper/handshake.go#L154-L247) is very similar to OpenTry:



Copy

func (k Keeper) ConnOpenAck(

ctx sdk.Context,

connectionID string,

clientState exported.ClientState, // client state for chain A on chain B

version \*types.Version, // version that Chain B chose in ConnOpenTry

counterpartyConnectionID string,

proofTry []byte, // proof that connectionEnd was added to Chain B state in ConnOpenTry

proofClient []byte, // proof of client state on chain B for chain A

proofConsensus []byte, // proof that chain B has stored ConsensusState of chain A on its client

proofHeight exported.Height, // height that relayer constructed proofTry

consensusHeight exported.Height, // latest height of chain A that chain B has stored on its chain A client

) ...

Both functions do the same checks, except that OpenTry takes proofInit as a parameter, and OpenAck takes proofTry:



Copy

// @ func (k Keeper) ConnOpenAck

// This function verifies that the snapshot we have of the counter-party chain looks like the counter-party chain, verifies the light client we have of the counter-party chain

// Check that Chain A committed expectedConnectionEnd to its state

if err := k.VerifyConnectionState(

ctx, connection, proofHeight, proofTry, counterparty.ConnectionId,

expectedConnection,

); err != nil {

return "", err

}

// This function verifies that the snapshot the counter-party chain has of us looks like us, verifies our light client on the counter-party chain

// Check that Chain A stored the clientState provided in the msg

if err := k.VerifyClientState(ctx, connection, proofHeight, proofClient, clientState); err != nil {

return "", err

}

// This function verifies that the snapshot the counter-party chain has of us looks like us, verifies our light client on the counter-party chain

// Check that Chain A stored the correct ConsensusState of chain B at the given consensusHeight

if err := k.VerifyClientConsensusState(

ctx, connection, proofHeight, consensusHeight, proofConsensus, expectedConsensusState,

); err != nil {

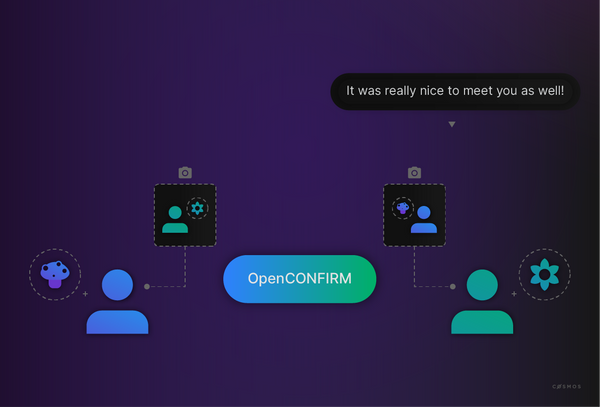
return "", err

}

Therefore, each chain verifies the ConnectionState, the ClientState, and the ConsensusState of the other chain. Note that after this step the connection state on chain A updates from INIT to OPEN.

[#Copy link](https://ida.interchain.io/academy/3-ibc/2-connections.html#handshake-step-4-connopenconfirm) Handshake step 4 - ConnOpenConfirm

OpenConfirm is the final handshake, in which chain B confirms that both self-identification and counterparty identification were successful.



The [conclusion of this handshake (opens new window)↗](https://github.com/cosmos/ibc-go/blob/v5.1.0/modules/core/03-connection/keeper/handshake.go#L253-L297) results in the successful establishing of an IBC connection:



Copy

func (k Keeper) ConnOpenConfirm(

ctx sdk.Context,

connectionID string,

proofAck []byte, // proof that connection opened on Chain A during ConnOpenAck

proofHeight exported.Height, // height that relayer constructed proofAck

)

The initiation of this handshake from chain B updates its connection state from TRYOPEN to OPEN. The counterparty chain *must* have an OPEN connection state in order for the handshake and connection state update to be successful.



The successful four-way handshake described establishes an IBC connection between the two chains.

Now consider two edge circumstances: simultaneous attempts by the chains to perform the same handshake, and attempts by an imposter to interfere.

[#Copy link](https://ida.interchain.io/academy/3-ibc/2-connections.html#crossing-hellos) Crossing hellos

"Crossing hellos" refers to when both chains attempt the same handshake step at the same time.



While still discussed in the video earlier, crossing hellos have been removed from ibc-go v4 onward, as referenced in [this PR (opens new window)↗](https://github.com/cosmos/ibc-go/pull/1672). The PreviousConnectionId in MsgConnectionOpenTry has been deprecated.

[#Copy link](https://ida.interchain.io/academy/3-ibc/2-connections.html#an-imposter) An imposter

What if an imposter tried to open a connection pretending to be another chain?

In fact this is not an issue. Any attempted OpenInit from an imposter will fail on OpenTry, because it will not contain valid proofs of Client/Connection/ConsensusState.

synopsis

To summarize, this section has explored:

* How a connection between two blockchains with IBC is established by a four-way handshake, thereby establishing the identity of the counterparty chain and preventing any malicious entity from pretending to be the counterparty.
* How versioning is important to establish, to ensure that only compatible protocol versions attempt to connect.

previous

[](https://ida.interchain.io/academy/3-ibc/1-what-is-ibc.html)

**[What is IBC?](https://ida.interchain.io/academy/3-ibc/1-what-is-ibc.html)**

up next

**[IBC/TAO - Channels (OPTIONAL)](https://ida.interchain.io/academy/3-ibc/3-channels.html)**

[[](https://ida.interchain.io/academy/3-ibc/3-channels.html)](https://ida.interchain.io/academy/3-ibc/3-channels.html)

Rate this Page

icon smile

icon meh

icon frown

Would you like to add a message?

Submit

Thank you for your Feedback!

On this page

[Connections](https://ida.interchain.io/academy/3-ibc/2-connections.html#connections)

[Version negotiation](https://ida.interchain.io/academy/3-ibc/2-connections.html#version-negotiation)

[Connection handshakes and states](https://ida.interchain.io/academy/3-ibc/2-connections.html#connection-handshakes-and-states)

[Handshake step 1 - ConnOpenInit](https://ida.interchain.io/academy/3-ibc/2-connections.html#handshake-step-1-connopeninit)

[Handshake step 2 - ConnOpenTry](https://ida.interchain.io/academy/3-ibc/2-connections.html#handshake-step-2-connopentry)

[Handshake step 3 - ConnOpenAck](https://ida.interchain.io/academy/3-ibc/2-connections.html#handshake-step-3-connopenack)

[Handshake step 4 - ConnOpenConfirm](https://ida.interchain.io/academy/3-ibc/2-connections.html#handshake-step-4-connopenconfirm)

[Crossing hellos](https://ida.interchain.io/academy/3-ibc/2-connections.html#crossing-hellos)

[An imposter](https://ida.interchain.io/academy/3-ibc/2-connections.html#an-imposter)

#### **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)