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

[Interchain Developer Academy](https://ida.interchain.io/)/[Interchain Developer Academy](https://ida.interchain.io/tutorials/1-tech-terms/)



Search

[Interchain Developer Academy](https://ida.interchain.io/)[Interchain Developer Academy](https://ida.interchain.io/tutorials/1-tech-terms/)

Search



Filters

Interchain Developer Academy

[](https://ida.interchain.io/tutorials/1-tech-terms/)

[Week 0 - Getting Started](https://ida.interchain.io/tutorials/1-tech-terms/)

[Getting Started](https://ida.interchain.io/tutorials/1-tech-terms/)

[Blockchain 101](https://ida.interchain.io/tutorials/1-tech-terms/)

[Blockchain History](https://ida.interchain.io/tutorials/1-tech-terms/)

[Public and Managed Blockchains](https://ida.interchain.io/tutorials/1-tech-terms/)

[Consensus in Distributed Networks](https://ida.interchain.io/tutorials/1-tech-terms/)

[Cryptography](https://ida.interchain.io/tutorials/1-tech-terms/)

[Self-Assessment Quiz](https://ida.interchain.io/tutorials/1-tech-terms/)

[Go Introduction - First Steps](https://ida.interchain.io/tutorials/1-tech-terms/)

[Go Basics](https://ida.interchain.io/tutorials/1-tech-terms/)

[Go Interfaces](https://ida.interchain.io/tutorials/1-tech-terms/)

[Control Structures in Go](https://ida.interchain.io/tutorials/1-tech-terms/)

[Arrays and Slices in Go](https://ida.interchain.io/tutorials/1-tech-terms/)

[Standard Packages in Go](https://ida.interchain.io/tutorials/1-tech-terms/)

[Concurrency in Go](https://ida.interchain.io/tutorials/1-tech-terms/)

[Good-To-Know Dev Terms](https://ida.interchain.io/tutorials/1-tech-terms/)

[Docker Introduction](https://ida.interchain.io/tutorials/1-tech-terms/)

[](https://ida.interchain.io/tutorials/1-tech-terms/)

[Week 1 - Introduction to the Interchain](https://ida.interchain.io/tutorials/1-tech-terms/)

[Introduction to the Interchain](https://ida.interchain.io/tutorials/1-tech-terms/)

[Blockchain Technology and the Interchain](https://ida.interchain.io/tutorials/1-tech-terms/)

[The Interchain Ecosystem](https://ida.interchain.io/tutorials/1-tech-terms/)

[Getting ATOM and Staking It](https://ida.interchain.io/tutorials/1-tech-terms/)

[A Blockchain App Architecture](https://ida.interchain.io/tutorials/1-tech-terms/)

[Accounts](https://ida.interchain.io/tutorials/1-tech-terms/)

[Transactions](https://ida.interchain.io/tutorials/1-tech-terms/)

[Messages](https://ida.interchain.io/tutorials/1-tech-terms/)

[Modules](https://ida.interchain.io/tutorials/1-tech-terms/)

[Protobuf](https://ida.interchain.io/tutorials/1-tech-terms/)

[Multistore and Keepers](https://ida.interchain.io/tutorials/1-tech-terms/)

[BaseApp](https://ida.interchain.io/tutorials/1-tech-terms/)

[Queries](https://ida.interchain.io/tutorials/1-tech-terms/)

[Events](https://ida.interchain.io/tutorials/1-tech-terms/)

[Context](https://ida.interchain.io/tutorials/1-tech-terms/)

[Testing](https://ida.interchain.io/tutorials/1-tech-terms/)

[Relaying with IBC](https://ida.interchain.io/tutorials/1-tech-terms/)

[Interchain Security](https://ida.interchain.io/tutorials/1-tech-terms/)

[Bridges](https://ida.interchain.io/tutorials/1-tech-terms/)

[Migrations](https://ida.interchain.io/tutorials/1-tech-terms/)

[Week 1 Quiz](https://ida.interchain.io/tutorials/1-tech-terms/)

[](https://ida.interchain.io/tutorials/1-tech-terms/)

[Week 2 - First Steps](https://ida.interchain.io/tutorials/1-tech-terms/)

[First Steps](https://ida.interchain.io/tutorials/1-tech-terms/)

[Setup Your Work Environment](https://ida.interchain.io/tutorials/1-tech-terms/)

[Run a Node, API, and CLI](https://ida.interchain.io/tutorials/1-tech-terms/)

[Ignite CLI](https://ida.interchain.io/tutorials/1-tech-terms/)

[Exercise - Make a Checkers Blockchain](https://ida.interchain.io/tutorials/1-tech-terms/)

[Store Object](https://ida.interchain.io/tutorials/1-tech-terms/)

[Create Custom Messages](https://ida.interchain.io/tutorials/1-tech-terms/)

[Create and Save a Game Properly](https://ida.interchain.io/tutorials/1-tech-terms/)

[Add a Way to Make a Move](https://ida.interchain.io/tutorials/1-tech-terms/)

[Emit Game Information](https://ida.interchain.io/tutorials/1-tech-terms/)

[Record the Game Winner](https://ida.interchain.io/tutorials/1-tech-terms/)

[Week 2 Exercise](https://ida.interchain.io/tutorials/1-tech-terms/)

[](https://ida.interchain.io/tutorials/1-tech-terms/)

[Week 3 - Introduction to IBC and CosmJS](https://ida.interchain.io/tutorials/1-tech-terms/)

[Introduction to IBC and CosmJS](https://ida.interchain.io/tutorials/1-tech-terms/)

[What is IBC?](https://ida.interchain.io/tutorials/1-tech-terms/)

[IBC/TAO - Connections (OPTIONAL)](https://ida.interchain.io/tutorials/1-tech-terms/)

[IBC/TAO - Channels (OPTIONAL)](https://ida.interchain.io/tutorials/1-tech-terms/)

[IBC/TAO - Clients (OPTIONAL)](https://ida.interchain.io/tutorials/1-tech-terms/)

[IBC Token Transfer](https://ida.interchain.io/tutorials/1-tech-terms/)

[Interchain Accounts (OPTIONAL)](https://ida.interchain.io/tutorials/1-tech-terms/)

[IBC Middleware (OPTIONAL)](https://ida.interchain.io/tutorials/1-tech-terms/)

[Create a Custom IBC Middleware (OPTIONAL)](https://ida.interchain.io/tutorials/1-tech-terms/)

[Integrate IBC Middleware Into a Chain (OPTIONAL)](https://ida.interchain.io/tutorials/1-tech-terms/)

[IBC Tooling](https://ida.interchain.io/tutorials/1-tech-terms/)

[What is CosmJS?](https://ida.interchain.io/tutorials/1-tech-terms/)

[Your First CosmJS Actions](https://ida.interchain.io/tutorials/1-tech-terms/)

[Compose Complex Transactions](https://ida.interchain.io/tutorials/1-tech-terms/)

[Learn to Integrate Keplr](https://ida.interchain.io/tutorials/1-tech-terms/)

[Create Custom CosmJS Interfaces](https://ida.interchain.io/tutorials/1-tech-terms/)

[](https://ida.interchain.io/tutorials/1-tech-terms/)

[Week 4 - Ignite CLI and IBC Advanced](https://ida.interchain.io/tutorials/1-tech-terms/)

[Ignite CLI and IBC Advanced](https://ida.interchain.io/tutorials/1-tech-terms/)

[Keep an Up-To-Date Game Deadline](https://ida.interchain.io/tutorials/1-tech-terms/)

[Keep Track Of How Many Moves Have Been Played](https://ida.interchain.io/tutorials/1-tech-terms/)

[Put Your Games in Order](https://ida.interchain.io/tutorials/1-tech-terms/)

[Auto-Expiring Games](https://ida.interchain.io/tutorials/1-tech-terms/)

[Let Players Set a Wager](https://ida.interchain.io/tutorials/1-tech-terms/)

[Handle wager payments](https://ida.interchain.io/tutorials/1-tech-terms/)

[Integration tests](https://ida.interchain.io/tutorials/1-tech-terms/)

[Incentivize Players](https://ida.interchain.io/tutorials/1-tech-terms/)

[Help Find a Correct Move](https://ida.interchain.io/tutorials/1-tech-terms/)

[Play With Cross-Chain Tokens](https://ida.interchain.io/tutorials/1-tech-terms/)

[Understand IBC Denoms](https://ida.interchain.io/tutorials/1-tech-terms/)

[Go Relayer](https://ida.interchain.io/tutorials/1-tech-terms/)

[Hermes Relayer](https://ida.interchain.io/tutorials/1-tech-terms/)

[](https://ida.interchain.io/tutorials/1-tech-terms/)

[Week 5 - CosmJS Advanced](https://ida.interchain.io/tutorials/1-tech-terms/)

[CosmJS Advanced](https://ida.interchain.io/tutorials/1-tech-terms/)

[Create Custom Objects](https://ida.interchain.io/tutorials/1-tech-terms/)

[Create Custom Messages](https://ida.interchain.io/tutorials/1-tech-terms/)

[Get an External GUI](https://ida.interchain.io/tutorials/1-tech-terms/)

[Integrate CosmJS and Keplr](https://ida.interchain.io/tutorials/1-tech-terms/)

[Backend Script for Game Indexing](https://ida.interchain.io/tutorials/1-tech-terms/)

[](https://ida.interchain.io/tutorials/1-tech-terms/)

[Week 6 - IBC Deep Dive](https://ida.interchain.io/tutorials/1-tech-terms/)

[IBC Deep Dive](https://ida.interchain.io/tutorials/1-tech-terms/)

[IBC Application Developer Introduction](https://ida.interchain.io/tutorials/1-tech-terms/)

[Make a Module IBC-Enabled](https://ida.interchain.io/tutorials/1-tech-terms/)

[Adding Packet and Acknowledgment Data](https://ida.interchain.io/tutorials/1-tech-terms/)

[Extend the Checkers Game With a Leaderboard](https://ida.interchain.io/tutorials/1-tech-terms/)

[Create a Leaderboard Chain](https://ida.interchain.io/tutorials/1-tech-terms/)

[](https://ida.interchain.io/tutorials/1-tech-terms/)

[Week 7 - From Code to MVP to Production and Migrations](https://ida.interchain.io/tutorials/1-tech-terms/)

[From Code to MVP to Production and Migrations](https://ida.interchain.io/tutorials/1-tech-terms/)

[Run in Production](https://ida.interchain.io/tutorials/1-tech-terms/)

[Prepare the Software to Run](https://ida.interchain.io/tutorials/1-tech-terms/)

[Prepare a Validator and Keys](https://ida.interchain.io/tutorials/1-tech-terms/)

[Prepare Where the Node Starts](https://ida.interchain.io/tutorials/1-tech-terms/)

[Prepare and Connect to Other Nodes](https://ida.interchain.io/tutorials/1-tech-terms/)

[Configure, Run, and Set Up a Service](https://ida.interchain.io/tutorials/1-tech-terms/)

[Prepare and Do Migrations](https://ida.interchain.io/tutorials/1-tech-terms/)

[Simulate Production in Docker](https://ida.interchain.io/tutorials/1-tech-terms/)

[Tally Player Info After Production](https://ida.interchain.io/tutorials/1-tech-terms/)

[Add a Leaderboard as a Module](https://ida.interchain.io/tutorials/1-tech-terms/)

[Migrate the Leaderboard Module After Production](https://ida.interchain.io/tutorials/1-tech-terms/)

[Simulate a Migration in Docker](https://ida.interchain.io/tutorials/1-tech-terms/)

[Final Exam](https://ida.interchain.io/tutorials/1-tech-terms/)

[](https://ida.interchain.io/tutorials/1-tech-terms/)

[What's Next?](https://ida.interchain.io/tutorials/1-tech-terms/)

[Continue Your Interchain Journey](https://ida.interchain.io/tutorials/1-tech-terms/)

Docs Version Switcher

On this page

[The Interchain](https://ida.interchain.io/tutorials/1-tech-terms/#the-interchain)

[The Interchain Stack](https://ida.interchain.io/tutorials/1-tech-terms/#the-interchain-stack)

[Cosmos Hub](https://ida.interchain.io/tutorials/1-tech-terms/#cosmos-hub)

[Light client daemon (LCD)](https://ida.interchain.io/tutorials/1-tech-terms/#light-client-daemon-lcd)

[Remote procedure call (RPC)](https://ida.interchain.io/tutorials/1-tech-terms/#remote-procedure-call-rpc)

[How does an RPC request work?](https://ida.interchain.io/tutorials/1-tech-terms/#how-does-an-rpc-request-work)

[RPC and the Interchain](https://ida.interchain.io/tutorials/1-tech-terms/#rpc-and-the-interchain)

[Protobuf](https://ida.interchain.io/tutorials/1-tech-terms/#protobuf)

[gRPC](https://ida.interchain.io/tutorials/1-tech-terms/#grpc)

[gRPC and Interchain](https://ida.interchain.io/tutorials/1-tech-terms/#grpc-and-interchain)

[gRPC-web](https://ida.interchain.io/tutorials/1-tech-terms/#grpc-web)

[gRPC-gateway](https://ida.interchain.io/tutorials/1-tech-terms/#grpc-gateway)

[Amino](https://ida.interchain.io/tutorials/1-tech-terms/#amino)

[#Copy link](https://ida.interchain.io/tutorials/1-tech-terms/#good-to-know-dev-terms) **Good-To-Know Dev Terms**

You can find an overview of several technical terms in this section, including an explanation of each term and links to further resources - all of which are essential when developing with the Interchain Stack and Cosmos SDK.



In this section, you will take a look at the following terms:

* The Interchain, and the Interchain Stack
* LCD
* RPC
* Protobuf - Protocol Buffers
* gRPC, gRPC-web, and gRPC-Gateway
* Amino

All these terms relate to how node interaction is conducted in Cosmos SDK blockchains.

Let's dive right into it.

[#Copy link](https://ida.interchain.io/tutorials/1-tech-terms/#the-interchain) The Interchain

**The Interchain** refers to *the network of application-specific blockchains* built with the Interchain Stack and inter-connected through the Inter-Blockchain Communication Protocol (IBC).



In case you stumble across the term Cosmos, please be aware that **the Interchain** used to be known as Cosmos. The terms "Cosmos", "Interchain Ecosystem", and "Interchain" can be understood as synonymous.

[#Copy link](https://ida.interchain.io/tutorials/1-tech-terms/#the-interchain-stack) The Interchain Stack

The various tools available to Interchain developers can be referred to collectively as the **Interchain Stack**.



Tools within the Interchain Stack, which contain "Cosmos" in their name will remain unchanged by current terminology changes, such as the **Cosmos SDK** and **CosmWasm**. Any chain built with the Cosmos SDK can typically be referred to as "a Cosmos chain" or "appchain".

[#Copy link](https://ida.interchain.io/tutorials/1-tech-terms/#cosmos-hub) Cosmos Hub

The Cosmos Hub is a chain that serves as an economic hub of the Interchain and service provider to other Cosmos chains. Built with the Interchain Stack, the Hub is home to the ATOM token, Interchain Security, and builders of the Cosmos SDK, CometBFT, and IBC.

[#Copy link](https://ida.interchain.io/tutorials/1-tech-terms/#light-client-daemon-lcd) Light client daemon (LCD)

A **light client**, compared to a full node, tracks only pieces of certain information on a blockchain. Light clients do not track the entire state of a blockchain and also do not contain every transaction/block of a chain.

In the Tendermint consensus, the light client protocol allows clients to benefit from the same degree of security that full nodes benefit from, while bandwidth requirements are minimized. A client can receive cryptographic proofs for blockchain states and transactions without having to sync all blocks or even their headers.



Take a look at [Light Clients in Tendermint Consensus (opens new window)↗](https://blog.cosmos.network/light-clients-in-tendermint-consensus-1237cfbda104) by Ethan Frey to discover more about how light clients are used in the Tendermint consensus.

Therefore, light clients are also vital to how the Inter-Blockchain Communication Protocol (IBC) can track information such as timestamps, root hashes, and the next validator set hash. This saves space and increases efficiency for state update processing.

The **light client daemon (LCD)** is an HTTP1.1 server exposed by the Cosmos SDK, and its default port is 1317. It exposes a REST API for the chain, meaning a *representational state transfer application programming interface* - this API allows for interaction with a RESTful web service. Traditionally, every query was re-implemented for LCD and routed to RPC behind the scenes.

****

**Why is it called light client daemon?**

Before SDK v0.40, to get a REST API it was necessary to run another backend service (or [*daemon (opens new window)*](https://en.wikipedia.org/wiki/Daemon_(computing)), a term inherited from Unix), for example using gaiacli rest-server --laddr 0.0.0.0:1317 --node localhost:26657. In Cosmos SDK v0.40, REST was moved inside the node service making it part of the Cosmos SDK, but the term "daemon" stuck, leading to the name *light client daemon* (LCD).

[#Copy link](https://ida.interchain.io/tutorials/1-tech-terms/#remote-procedure-call-rpc) Remote procedure call (RPC)

A **remote procedure call (RPC)** is *a software communication protocol*. The term is often found in distributed computing because RPC is a technique to realize inter-process communication (IPC) by allowing a program to cause a subroutine procedure that is executed in a different address space (a different machine).

RPC can be understood as a client-server interaction in which the "caller" is the client, more specifically the requesting program, and the "executor" is the server, more specifically the service-providing program. The interaction is implemented through a request-response message-passing system.

In short, RPC is a request-response protocol, initiated by a client sending a request to a remote server to execute the subroutine.

RPC allows calling functions in different address spaces. Usually, the called functions are run on a different computer than the one calling them. However, with RPC, the developer codes as if the subroutine would be local; the developer does not have to code in details for remote interaction. Thus, with RPCs it is implied that all calling procedures are basically the same, independent of them being local or remote calls.



As RPCs implement remote request-response protocols, it is important to note that remote procedure calls can fail in case of network problems.

[#Copy link](https://ida.interchain.io/tutorials/1-tech-terms/#how-does-an-rpc-request-work) How does an RPC request work?

In general, when a remote procedure call is invoked the procedure parameters are transferred across the network to the execution environment where the procedure is executed. Once finished, the results of the procedure call invoked are transferred to the call environment. The execution then resumes in the call environment, just as it would in a regular local procedure call.

A step-by-step RPC request could look like the following:

1. A client calls a client stub - a piece of code converting parameters that are passed between client and servers during an RPC. The call is a local procedure call.



A stub is a small program routine substituting a longer program. This allows machines to behave as if a program on a remote machine was operating locally. The client has a stub that interfaces with the remote procedure, while the server has a stub to interface with the original request procedure.

In RPCs, the client's stub substitutes for the program providing a request procedure. The stub accepts and forwards the request to the remote procedure. Once the remote procedure completes the request, it returns the results to the stub which in turn passes them to the request procedure.

The server also has a stub to interface with the remote procedure.

1. The client stub packs the procedure parameters into a message.



Packing procedure parameters is called **marshaling**.

Specifically, this is *the process of gathering data from one or more applications, putting data pieces into a message buffer, and organizing the data into a prescribed data format*.

Marshaling is vital to pass output parameters of a program written in one language as inputs to programs in a different language.

1. The client stub then makes a system call to send the message.
2. The client's local operating system (OS) sends the message from the client (machine A) to the server (machine B) through the corresponding transport layers.
3. The server OS passes the incoming packets to the server stub.
4. The server stub unpacks the message and with it the included procedure parameters - this is called **unmarshaling**.
5. The server stub calls a server procedure and the procedure is executed.
6. Once the procedure is finalized, the output is returned to the server stub.
7. The server stub packs the return values into a message.
8. The message is sent to the transport layer, which sends the message to the client's transport layer.
9. The client stub unmarshals the return parameters and returns them to the original calling client.



In an Open Systems Interconnection (OSI) model, RPC touches the transport and application layers.

The transport layer is tasked with the reliable sending and receiving of messages across a network. It requires error-checking mechanisms, data flow controls, data integrity assurance, congestion avoidance, multiplexing, and same order delivery.

The application layer is tasked with ensuring effective communication between applications on different computer systems and networks. It is a component of an application that controls methods of communication with other devices.

[#Copy link](https://ida.interchain.io/tutorials/1-tech-terms/#rpc-and-the-interchain) RPC and the Interchain

In the Interchain Stack, RPCs are used by the command-line interface (CLI) among other things to access chains. A node exposes several endpoints - gRPC, REST, and CometBFT endpoint.

Exposed by CometBFT, the CometBFT RPC endpoint is an HTTP1.1 server. The default port is 26657. The gRPC server's default port is 9090, and the REST server's default port is 1317. The CometBFT RPC is independent of the Cosmos SDK and can be configured. It uses HTTP POST and JSON-RPC 2.0 for data encoding.



For more information on the CometBFT RPC, gRPC, and the REST server, a closer look at the [Cosmos SDK documentation (opens new window)↗](https://docs.cosmos.network/main/core/grpc_rest.html) is recommended.



The Interchain Stack exposes both the CometBFT RPC and the LCD. For example, [CosmJS](https://ida.interchain.io/tutorials/7-cosmjs/1-cosmjs-intro.html) uses RPC to implement a JSON-RPC API.

[#Copy link](https://ida.interchain.io/tutorials/1-tech-terms/#protobuf) Protobuf

**Protobuf** (for "Protocol Buffers") is an open-source, cross-platform data format developed by Google. It helps serialize structured data and assists with program communication in networks or when storing data.



If you want to get more accustomed to Protobuf, a look at the [official documentation (opens new window)↗](https://developers.google.com/protocol-buffers) helps dive deeper and offers guides and tutorials.

Also, take a look at the [section on this platform on Protobuf](https://ida.interchain.io/academy/2-cosmos-concepts/6-protobuf.html).

In the Interchain Stack, Protobuf is a data serialization method that developers use to describe message formats. There is a lot of internal communication within an Interchain application, and Protobuf is central to how communication is done.

With Cosmos SDK v0.40, Protobuf began replacing Amino as the data encoding format of chain states and transactions, in part because encoding/decoding performance is better with Protobuf than Amino. In addition, the developer tooling is also better for Protobuf. Another benefit of switching is that the use of gRPC is fostered, as Protobuf automatically defines and generates gRPC functions. Thus developers no longer have to implement the same query for RPC, LCD, and CLI.

[#Copy link](https://ida.interchain.io/tutorials/1-tech-terms/#grpc) gRPC

**gRPC** is *an open-source, high-performance remote procedure call (RPC) framework*. It was developed by Google to handle RPCs and released in 2016. gRPC can run in any environment and supports a variety of programming languages.



For more on gRPC and very helpful information on getting started, take a look at the [gRPC documentation (opens new window)↗](https://grpc.io/).

gRPC uses HTTP2 for transport and Protocol Buffers (Protobuf) to encode data. gRPCs have a single specification, which makes all gRPC implementations consistent.

[#Copy link](https://ida.interchain.io/tutorials/1-tech-terms/#grpc-and-interchain) gRPC and Interchain

In the Interchain Stack, gRPCs are transmission control protocol (TCP) servers with Protobuf and are used for data encoding. The default port is 9090.



Transmission control protocol (TCP) is one of the main internet protocols that allows establishing a connection between a client and server to send data. TCP makes communication between application programs and the internet protocol (IP) possible.

In the Cosmos SDK, Protobuf is the main encoding library.

****

**Encoding in the Cosmos SDK**

A **wire encoding protocol** is *a protocol defining how data is transported from one point to another*. Wire protocols describe ways in which information is exchanged at the application level. Thus it is a communication protocol of the application layer protocol and not a transport protocol. To define the data exchange, the wire protocol requires specific attributes regarding:

* Data types - units of data, message formats, etc.
* Communication endpoints
* Capabilities - delivery guarantees, direction of communication, etc.

Wire protocols can be text-based or binary protocols.

In the Cosmos SDK, there are **two categories of binary wire encoding types**: client encoding, and store encoding. Whereas client encoding deals with transaction processing and signing transactions, store encoding tackles state-machine transactions and with it what is stored in the Merkle tree.

The Cosmos SDK uses two binary wire encoding protocols:

* **Amino:** an object encoding specification. Every Cosmos SDK module uses an Amino codec to serialize types and interfaces.
* [Protocol Buffers (Protobuf)](https://ida.interchain.io/academy/2-cosmos-concepts/6-protobuf.html)**:** a data serialization method, which developers use to describe message formats.

Due to reasons such as performance drawbacks and missing cross-language/client support, Protocol Buffers are used more and more over Amino.

For more information on encoding in the Cosmos SDK, see the [Cosmos SDK documentation (opens new window)](https://docs.cosmos.network/main/core/encoding.html).

[#Copy link](https://ida.interchain.io/tutorials/1-tech-terms/#grpc-web) gRPC-web

gRPC is supported across different software and hardware platforms. **gRPC-web** is *a JavaScript implementation of gRPC for browser clients*. gRPC-web clients connect to gRPC services via a special proxy.



For more on gRPC-web, a closer look at the [gRPC repository (opens new window)↗](https://github.com/grpc/grpc-web) is recommended.

To dive into developing with gRPC-web, the [documentation's quick start (opens new window)↗](https://grpc.io/docs/platforms/web/quickstart/) and [basics tutorials (opens new window)↗](https://grpc.io/docs/platforms/web/basics/) are very valuable resources.

As with gRPC in general, gRPC-web uses HTTP2 with Protobuf for data encoding. The default port is 9091.



Secret.js is a JavaScript SDK used to write applications interacting with the [Secret Network (opens new window)↗](https://scrt.network/), which uses gRPC-web.

[#Copy link](https://ida.interchain.io/tutorials/1-tech-terms/#grpc-gateway) gRPC-gateway

**gRPC-gateway** is a tool to expose gRPC endpoints as REST endpoints. It helps provide APIs in gRPC and RESTful style, reads gRPC service definitions, and generates reverse-proxy servers that can translate a RESTful JSON API into gRPC. For each gRPC endpoint defined in a Protobuf Query service, the Cosmos SDK offers a corresponding REST endpoint.

gRPC-Gateway's aim is ["to provide that HTTP+JSON interface to your gRPC service" (opens new window)↗](https://grpc-ecosystem.github.io/grpc-gateway/docs/overview/background/). With it, developers can benefit from all the advantages of gRPC and, at the same time, still provide a RESTful API - a very helpful tool when for example you want to develop a web application but have browsers that do not support HTTP2. This can help ensure backward compatibility, and multi-language, multi-client support.



If you want to explore gRPC-Gateway, a closer look at the [gRPC-Gateway documentation (opens new window)↗](https://grpc-ecosystem.github.io/grpc-gateway/) is recommended.

In the Cosmos SDK, gRPC-Gateway provides an HTTP1.1 server with REST API and a base64-encoded Protobuf for data encoding; it exposes gRPC endpoints as REST endpoints. It routes on the server to gRPC and piggybacks off of LCD, thus it is also on port 1317.

For example, if you cannot use gRPC for your application because a browser does not support HTTP2, you can still use the Cosmos SDK. The SDK provides REST routes via gRPC-Gateway.

[#Copy link](https://ida.interchain.io/tutorials/1-tech-terms/#amino) Amino

Amino is an object encoding specification. In the Cosmos SDK, every module uses an **Amino codec** that helps serialize types and interfaces. Amino handles interfaces by prefixing bytes before concrete types.

Usually, Amino codec types and interfaces are registered in the module's domain.



A concrete type is a non-interface type that implements a registered interface. Types need to be registered when stored in interface type fields, or in a list with interface elements.

As a best practice, upon initialization make sure to:

* Register the interfaces.
* Implement concrete types.
* Check for issues, like conflicting prefix bytes.

Every module exposes a function, RegisterLegacyAminoCodec. With it, users can provide a codec and register all types. Applications call this method for necessary modules.

With Amino, raw wire bytes are encoded and decoded to concrete types or interfaces when there is no Protobuf-based type definition for a module.



For more on Amino specifications and implementation for Go, see the [Tendermint Go Amino documentation (opens new window)↗](https://github.com/tendermint/go-amino).



Amino is basically JSON with some modifications. For example, the JSON specification does not define numbers greater than 2^53, so instead strings are used in Amino when encoding a type greater than uint64/int64.

For more on the Amino types and their representation in JSON, see the [Secret.js documentation (opens new window)↗](https://github.com/scrtlabs/secret.js/blob/master/DEVELOPERS.md#amino-types-and-how-theyre-represented-in-json).

previous

[](https://ida.interchain.io/tutorials/4-golang-intro/7-concurrency.html)

**[Concurrency in Go](https://ida.interchain.io/tutorials/4-golang-intro/7-concurrency.html)**

up next

**[Docker Introduction](https://ida.interchain.io/tutorials/5-docker-intro/)**

[[](https://ida.interchain.io/tutorials/5-docker-intro/)](https://ida.interchain.io/tutorials/5-docker-intro/)

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

[The Interchain](https://ida.interchain.io/tutorials/1-tech-terms/#the-interchain)

[The Interchain Stack](https://ida.interchain.io/tutorials/1-tech-terms/#the-interchain-stack)

[Cosmos Hub](https://ida.interchain.io/tutorials/1-tech-terms/#cosmos-hub)

[Light client daemon (LCD)](https://ida.interchain.io/tutorials/1-tech-terms/#light-client-daemon-lcd)

[Remote procedure call (RPC)](https://ida.interchain.io/tutorials/1-tech-terms/#remote-procedure-call-rpc)

[How does an RPC request work?](https://ida.interchain.io/tutorials/1-tech-terms/#how-does-an-rpc-request-work)

[RPC and the Interchain](https://ida.interchain.io/tutorials/1-tech-terms/#rpc-and-the-interchain)

[Protobuf](https://ida.interchain.io/tutorials/1-tech-terms/#protobuf)

[gRPC](https://ida.interchain.io/tutorials/1-tech-terms/#grpc)

[gRPC and Interchain](https://ida.interchain.io/tutorials/1-tech-terms/#grpc-and-interchain)

[gRPC-web](https://ida.interchain.io/tutorials/1-tech-terms/#grpc-web)

[gRPC-gateway](https://ida.interchain.io/tutorials/1-tech-terms/#grpc-gateway)

[Amino](https://ida.interchain.io/tutorials/1-tech-terms/#amino)

#### **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)