This design doc will help guide developers who want to integrate an execution environment to Celestia's Data Availability layer.

An example execution environment could be the EVM, Substrate, Move, or Solana.

Considerations

At first, you need to ask yourself: how much integration do you really want in order to connect to Celestia's Data Availability Layer.

The reason you need to ask this question is because it will inform the design considerations of your integration of the execution environment with Celestia's Data Availability layer.

You can look at multiple different approaches in order to determine how much integration work you want to do.

Optimint

Optimint is a Celestia library that allows for building ABCI rollups that connect directly to the Data Availability layer.

At the moment, Optimint is compatible with any ABCI applications that want to connect to Celestia's Data Availability layer.

One approach you can take here for integration your execution environment into Celestia is to extend Optimint to include the new execution environment. You will basically create an ABCI-wrapper around the VM of your execution environment and integrate that into Optimint.

A good example of this is ABCI-wrapper around Narwhal and Bullshark built by Paradigm. You can read the blog post here and check out the Github repo here.

It is important to consider here that for an existing execution layer, your main focus should be on integrating the VM only. Let's look at Geth, the go-ethereum client for EVM networks like Ethereum. You would want to integrate just the EVM code inside Geth. Otherwise, you will be increasing the complexity of your application, as you need to worry about the following things:

- Creating the block
- Propagating transactions
- Maintaining mempool
- Peer discovery
- Execution of transactions by ABCI
- Abstraction layer on top of ABCI
- Sequencer replacement to Consensus

By just focusing on building the ABCI-wrapper around Geth's EVM codebase, you don't have to worry so much about the other components listed above as your Execution Environment becomes wrapped in as an ABCI component.

An example of Geth working with Tendermint as an example is found in this Github repo here.

Non-ABCI Route with Optimint

You can technically just use Optimint directly with your Execution Environment without having to use an ABCI-wrapper around the Execution Environment. This is done by replacing the ABCI-portion of a modified Optimint with your Execution Environment. This would require forking Optimint and is also considered the hardest route to take on your integration steps to plug-in to Celestia's Data Availability layer.

The long-term goals of Optimint are to make it a very modular library to allow for more Execution Environments to plug-in to it with it's own dedicated API. Other future goals will make it a library for rollups and a way of introducing VMs or execution layers on top of Optimint.

One benefit of following this harder route is by doing this, such work can help inform the Optimint team better on what an Optimint API would look like.

Modify the Node of the Execution Environment

The other route you can take is to modify the node codebase of your execution environment so that it can submit blocks

directly to the Celestia Data Availability Layer. We currently have go-cnc, an API library that allows you to do this.

In this integration route, you will focus on submitting the block data to specific namespace ID on the DA layer.

An example reference code of how Optimint uses go-cnc is foundhere.