Cosmos Avail DA module

What is CADA (Cosmos Avail DA module)

CADA is a module designed to connect Cosmos sovereign chains with the Avail network, making it easier for any Cosmos chain or rollapp to use Avail as their Data Availability (DA) layer.

With CADA, developers can improve the scalability and security of their decentralized applications within the Cosmos ecosystem. It enables better data handling and availability, allowing Cosmos-based chains to tap into the strengths of Avail and build a more connected and resilient blockchain network.

Learn more about the CADA module below:

Cosmos Avail DA Module

What is Avail DA?

Avail is the permisionless unification layer for web3. Avail is designed to be a platform that connects different ecosystems by providing a modular, scalable, and interoperable platform. Avail's vision is to provide a cohesive, unified user experience within a flexible and modular blockchain ecosystem, drawing on lessons from Web2 to innovate in Web3.

With Avail's foundational DA layer (Avail DA), different ecosystems can innovate on top freely, while leveraging Nexus for cross-ecosystem messaging.

Learn more aboutAvail here .

Integration Guide

To integrate the CADA module into your Cosmos SDK application, follow the steps outlined in theintegration guide(opens in a new tab)

Note: Ensure that the Avail light client URL is correctly configured for the module to function as expected. For instructions on setup Avail locally, please refer tothis documentation(opens in a new tab) . You can also use the references below for integration.You can also simply run the module from these repositories as well:

Cosmos SDK Integration Example Avail DA Module with Simapp If you are running the chain from the Simapp repository, you can use thisinit-chain script(opens in a new tab) to set up the testnet.

How It Works

CADA (Cosmos Chain)

Initiates the process by running thePreBlocker ABCI method.

Request to Relayer

Sendsblock range information to the relayer.

Relayer

Fetches theblock data from the Cosmos Provider and posts it to the Avail Light Client.

Avail Light Client

Confirms whether the data is available.

IfYes : Broadcast the Avail height and status.

IfNo : Retry data submission.

Validators

Vote to confirm the data availability, updating the blob status to "Success" or "Failure" based on results. These are main components in the workflow:

1. CADA

The core functionality of theCADA module is integrated with and operates on the Cosmos blockchain.

In the CADA module:

- At each block interval, thePreBlocker
- ABCI method sends a request to theRelayer
- , specifying the range of block heights that are ready to be posted to theAvail

- Data Availability (DA) network.

- The chain is responsible for aggregating vote extensions from all validators and verifying whether the data has been made available on Avail.

- Since verification requires communicating with the light client, an asynchronous voting mechanism is needed.Vote extensions
- enable this asynchronous voting mechanism for verification purposes.

2. Relayer

TheRelayer facilitates communication between the Cosmos Chain, the Avail light client, and the Cosmos Provider.

- Data Submission
- : The relayer is responsible for fetching block data from the Cosmos provider and posting it to the Avail light client via an HTTP request.
- Based on the response from the light client, the relayer submits a transaction informing the validators of the data availability status and the specific Avail block height where the data is included, so that validators can verify it.
- Data Verification
- : During verification, the relayer communicates with the Avail light client to confirm whether the data is truly available at the specified height.

3. Avail Light Node

TheAvail Light Client allows interaction with the Avail DA network without requiring a full node, and without having to trust remote peers. It leveragesData Availability Sampling (DAS) , which the light client performs on every newly created block.

- The chain communicates with the Avail light client via the relayer during the data submission and data availability verification processes.

Find more details about the Avail Light Clienthere(opens in a new tab) .

4. Cosmos Provider

TheCosmos Provider is responsible for fetching block data via RPC so that the data can be posted to Avail for availability checks.

Workflow

- At each block interval, a request is sent from thePreBlocker
- ABCI method to the Keeper, specifying the range of block heights that are ready to be posted to theAvail
- DA network.
- The range of block heights should be fromprovenHeight + 1
- tomin(provenHeight + MaxBlocksLimitForBlob, CurrentBlockHeight)
- .
- If the status of the previous blocks is eitherSUCCESS
- orFAILURE
- , the status can be updated toPENDING
- .
- range = [fromBlock, toBlock] // (fromBlock < toBlock < CurrentBlock)
- status = PENDING
- TheProposer
- of the block will make a request to theRelayer
- to post the blocks data by passing the range of blocks to be posted.
- TheRelayer
- fetches the blocks data from the local provider, converts the blocks data to bytes, and posts that data toAvail
- .
- Once the success of data availability is confirmed, theRelayer
- broadcasts theAvail height
- at which the blob data is made available using theMsgUpdateBlobStatus
- transaction.

- The status, Avail height, and voting deadline will be updated in the state.
- status = IN_VOTING
- availHeight = tx.availHeight
- votingEndBlock = currentBlock + votingInterval
- At block heightVotingEndBlock - 1
- , all the validators verify if the specified blocks data is truly made available at the specified Avail height. They cast their vote (YES or NO) usingvote extensions
- .
- At block heightVotingEndBlock
- , all the votes fromvote_extensions
- will be collected and aggregated. If the collectivevoting power is > 66%
- , the status will be updated
- status = SUCCESS // success and ready for next blocks
- provenHeight = Range EndHeight // End Height from the given block range
- In case of failure at any stage, the whole flow will be repeated.

Paima Engine VectorX bridge