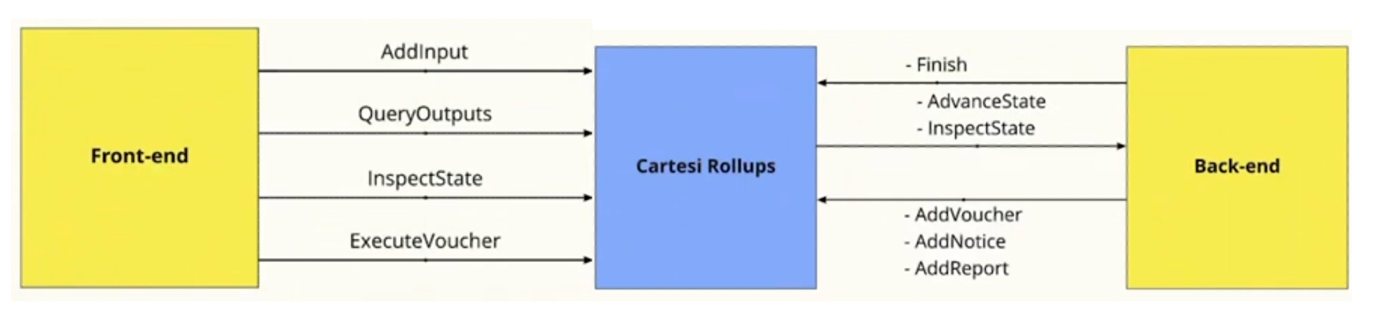
**Cartesi**



Rollup to Backend

* **Two type of Inputs:**
  + **Advance State:** Advances the state of machine
  + **Inspect State:** Inspect state of machine in Read Only Mode (Function always revert back computation to initial state)
* **Three types of outputs:**
  + **Add Voucher:** A way to send transactions to Ethereum for withdrawing assets or similar operations (Claimable after 7 days)
  + **Add Notice:** Equivalent to Ethereum events but with proof attached i.e. advances the state
  + **Add Report:** Equivalent to Logs i.e. has no proof attached i.e. the state isn’t advanced

Frontend to Rollup

* **Add Input:** All the advance state requests go through the rollup contract on L1 chain. You need to call *addInput* method on *inputBox* contract with address of the Dapp and Hex encoded request data.  
  cast send 0xinput\_box "addInput(dapp\_address, hex\_encoded\_payload)"
* **Query Outputs:** Query current state like Vouchers, Notices and Reports are done through GraphQL API
* **Inspect State:** To send inspect state request directly to the rollup. The *“mypath”* is the url encoded string request data i.e. you take the request string data (stringify it if its JSON) and then URL encode it. *curl http://localhost:5005/inspect/mypath*
* **Execute Voucher:** Execution of vouchers is done by sending request to rollup contract on L1 chain using a transaction after the wait period is over.

Handling ERC20 & ERC721

Now that we know how to handle generic payloads in our DApp, it is important to mention the concept of handling standardized Solidity interfaces, specifically ERC20 and ERC721 assets. To receive this special kind of *input data*, the Cartesi Rollup Framework provides a set of contracts called the [Portals](https://docs.cartesi.io/cartesi-rollups/components/#portals), which are specifically designed to [handle assets](https://docs.cartesi.io/cartesi-rollups/assets-handling/) on behalf of your DApp.

Asset handling in Cartesi DApps involves the following procedures:

1. Locking assets on the base layer by calling deposit methods on special contracts called [Portals](https://docs.cartesi.io/cartesi-rollups/components/#portals), which will effectively transfer asset ownership to the target DApp contract. There are specific portals for each kind of asset (Ether, ERC-20, ERC-721, ERC-1155).
2. The Cartesi Rollups framework notifies the DApp back-end of the deposit by sending it a special input.
3. The DApp's back-end code needs to recognize and handle the special input, in order to process the deposit according to its own logic (e.g., by storing each user's balance in a hash table or database).
4. When appropriate (e.g., when a game ends and the winner wishes to withdraw their funds), the back-end generates a [voucher](https://docs.cartesi.io/cartesi-rollups/components/#vouchers) that encodes a transfer of assets on the base layer, from the DApp to the target user. The actual withdrawal will take effect on the base layer when the voucher is executed. This is a secure process because it can only be done when the voucher has an associated validity proof ensuring that the validator nodes have reached consensus about its contents.