Precompiles

- Overview
- Available Precompiles
  - IsConfidential
  - ConfidentialInputs
  - ConfidentialStore
  - ConfidentialRetrieve
  - NewDataRecord
  - FetchDataRecords
  - EthCall
  - SimulateBundle
  - ExtractHint
  - SubmitBundleJsonRPC
  - FillMevShareBundle
  - BuildEthBlock
  - SubmitEthBlockBidToRelay
  - SignEthTransaction
- Precompiles Governance

Overview

Precompile are MEVM contracts that are implemented in native code instead of bytecode. Precompiles additionally can communicate with internal APIs. Currently the MEVM supports all existing Ethereum Precompiles up to Dencun, and introduces four new classes of precompiles:

1. offchain computation that is too expensive in solidity
2. calls to API methods to interact with the Confidential Data Store
3. calls tosuavex
4. API Methods to interact with Domain-Specific Services
5. calls to retrieve context for the confidential compute requests

Available Precompiles

A list of available precompiles in Rigil are as follows:

IsConfidential

Implementation

Address:0x0000000000000000000000000000000042010000

Determines if the current execution mode is regular (onchain) or confidential. Outputs a boolean value.

function

isConfidential ( )

internal

view

returns

(bool b)

ConfidentialInputs

Implementation

Address:0x0000000000000000000000000000000042010001

Provides the confidential inputs associated with a confidential computation request. Outputs are in bytes format.

function

confidentialInputs ( )

internal

view

returns

(bytes

memory)

ConfidentialStore

Implementation

Address:0x0000000000000000000000000000000042020000

Handles the storage of data in the confidential store. Requires the caller to be part of theAllowedPeekers for the associated bid.

function

confidentialStore (DataId dataId ,

string

memory key ,

bytes

memory data1)

internal

view

ConfidentialRetrieve

Implementation

Address:0x0000000000000000000000000000000042020001

Retrieves data from the confidential store. Also mandates the caller's presence in theAllowedPeekers list.

function

confidentialRetrieve (DataId dataId ,

string

memory key)

internal

view

returns

(bytes

memory)

NewDataRecord

Implementation

Address:0x0000000000000000000000000000000042030000

Initializes data records within the ConfidentialStore.AllowedPeekers specifies which addresses can "get" data.AllowedStores specifies which addresses can "set" data. Prior to storing data, all bids should undergo initialization via this precompile.

function

newDataRecord (uint64 decryptionCondition ,

address [ ]

memory allowedPeekers ,

address [ ]

memory allowedStores ,

string

memory dataType)

FetchDataRecords

Implementation

Address:0x0000000000000000000000000000000042030001

Retrieves all data records correlating with a specified decryption condition.

function

fetchDataRecords (uint64 cond ,

string

memory namespace)

internal

view

returns

(DataRecord [ ]

memory)

EthCall

Implementation

Address:0x0000000000000000000000000000000042100003

Uses theeth_call JSON RPC method to let you simulate a function call and return the response.

function

ethcall (address contractAddr ,

bytes

memory input1)

internal

view

returns

(bytes

memory)

SimulateBundle

Implementation

Address:0x0000000000000000000000000000000042100000

Performs a simulation of the bundle by building a block that includes it. Outputs indicate if the execution was successful and the Effective Gas Price of the resultant block.

function

simulateBundle (bytes

memory bundleData)

internal

view

returns

(uint64)

ExtractHint

Implementation

Address:0x0000000000000000000000000000000042100037

Interprets the bundle data and extracts hints, such as the "To" address and calldata.

function

extractHint (bytes

memory bundleData)

internal

view

returns

(bytes

memory)

SubmitBundleJsonRPC

Implementation

Address:0x0000000000000000000000000000000043000001

Submits bytes as JSONRPC message to the specified URL with the specified method. As this call is intended for bundles, it also signs the params and addsX-Flashbots-Signature header, as usual with bundles. Regular eth bundles don't need any processing to be sent.

function

submitBundleJsonRPC (string

memory url ,

string

memory method ,

bytes

memory params)

internal

view

returns

(bytes

memory)

FillMevShareBundle

Implementation

Address:0x0000000000000000000000000000000043200001

Joins the user's transaction and with the backrun, and returns encoded mev-share bundle. The bundle is ready to be sent viaSubmitBundleJsonRPC .

function

fillMevShareBundle (DataId dataId)

internal

view

returns

(bytes

memory)

BuildEthBlock

Implementation

Address:0x0000000000000000000000000000000042100001

Constructs an Ethereum block based on the providedbidIds . The construction follows the order ofbidId s are given .

function

buildEthBlock (BuildBlockArgs memory blockArgs , DataId dataId ,

string

memory namespace)

SubmitEthBlockBidToRelay

Implementation

Address:0x0000000000000000000000000000000042100002

Submits a given builderBid to a mev-boost relay. Outputs any errors that arise during submission.

function

submitEthBlockBidToRelay (string

memory relayUrl ,

bytes

memory builderBid)

SignEthTransaction

Implementation

Address:0x0000000000000000000000000000000040100001

Signs an Ethereum Transaction, 1559 or Legacy, and returns raw signed transaction bytes.txn is binary encoding of the transaction.signingKey is hex encoded string of the ECDSA private keywithout the 0x prefix .chainId is a hex encoded stringwith 0x prefix .

function

signEthTransaction (bytes

memory txn ,

string

memory chainId ,

string

memory signingKey)

view

returns

(bytes

memory)

Precompiles Governance

The governance process for adding precompiles is in it's early stages but is as follows:

- Discuss the idea in aforum post
- Open a PR and provide implementation
- Feedback and review
- Possibly merge and deploy in the next network upgrade, or sooner, depending on the precompileEdit this page

Previous MEVM Next Confidential Data Store