Precompiles

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 Toliman are as follows:

IsConfidential

Address:0x0000000000000000000000000000000042010000

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

function

isConfidential ( )

internal

view

returns

(bool b)

buildEthBlock

Address:0x0000000000000000000000000000000042100001

Constructs an Ethereum block based on the provided data records. No blobs are returned.

function

buildEthBlock (BuildBlockArgs memory blockArgs , DataId dataId ,

string

memory relayUrl)

internal

view

returns

(bytes

memory ,

bytes

memory) Inputs:

- blockArgs
- (BuildBlockArgs): Arguments to build the block
- dataId
- (DataId): ID of the data record with mev-share bundle data
- relayUrl
- (string): If specified the built block will be submitted to the relay

Outputs:

- blockBid
- (bytes): Block Bid encoded in JSON
- executionPayload
- (bytes): Execution payload encoded in JSON

buildEthBlockTo

Address:0x0000000000000000000000000000000042100006

Constructs an Ethereum block based on the provided data records. No blobs are returned.

function

buildEthBlockTo (string

memory executionNodeURL , BuildBlockArgs memory blockArgs , DataId dataId ,

string

memory relayUrl)

internal

view

returns

(bytes

memory ,

bytes

memory) Inputs:

- executionNodeURL
- (string): URL (or service name) of the execution node
- blockArgs
- (BuildBlockArgs): Arguments to build the block
- dataId
- (DataId): ID of the data record with mev-share bundle data
- relayUrl
- (string): If specified the built block will be submitted to the relay

Outputs:

- blockBid
- (bytes): Block Bid encoded in JSON
- executionPayload
- (bytes): Execution payload encoded in JSON

confidentialInputs

Address:0x0000000000000000000000000000000042010001

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

function

confidentialInputs ( )

internal

view

returns

(bytes

memory) Outputs:

- confindentialData
- (bytes): Confidential inputs

confidentialRetrieve

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) Inputs:

- dataId
- (DataId): ID of the data record to retrieve
- key
- (string): Key slot of the data to retrieve

Outputs:

- value
- (bytes): Value of the data

confidentialStore

Address:0x0000000000000000000000000000000042020000

Stores data in the confidential store. Requires the caller to be part of theAllowedPeekers for the associated data record.

function

confidentialStore (DataId dataId ,

string

memory key ,

bytes

memory value)

internal

view

returns

( ) Inputs:

- dataId
- (DataId): ID of the data record to store
- key
- (string): Key slot of the data to store
- value
- (bytes): Value of the data to store

contextGet

Address:0x0000000000000000000000000000000053300003

Retrieves a value from the context

function

contextGet (string

memory key)

internal

view

returns

(bytes

memory) Inputs:

- key
- (string): Key of the value to retrieve

Outputs:

- value
- (bytes): Value of the key

doHTTPRequest

Address:0x0000000000000000000000000000000043200002

Performs an HTTP request and returns the response.request is the request to perform.

function

doHTTPRequest (HttpRequest memory request)

internal

view

returns

(bytes

memory) Inputs:

- request
- (HttpRequest): Request to perform

Outputs:

- httpResponse
- (bytes): Body of the response

ethcall

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) Inputs:

- contractAddr
- (address): Address of the contract to call
- input1
- (bytes): Data to send to the contract

Outputs:

- callOutput
- (bytes): Output of the contract call

extractHint

Address:0x0000000000000000000000000000000042100037

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

function

extractHint (bytes

memory bundleData)

internal

view

returns

(bytes

memory) Inputs:

- bundleData
- (bytes): Bundle object encoded in JSON

Outputs:

- hints
- (bytes): List of hints encoded in JSON

fetchDataRecords

Address:0x0000000000000000000000000000000042030001

Retrieves all data records correlating with a specified decryption condition and namespace

function

fetchDataRecords (uint64 cond ,

string

memory namespace)

internal

view

returns

(DataRecord [ ]

memory) Inputs:

- cond

- (uint64): Filter for the decryption condition
- namespace
- (string): Filter for the namespace of the data records

Outputs:

- dataRecords
- (DataRecord[]): List of data records that match the filter

fillMevShareBundle

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) Inputs:

- dataId
- (DataId): ID of the data record with mev-share bundle data

Outputs:

- encodedBundle
- (bytes): Mev-Share bundle encoded in JSON

newBuilder

Address:0x0000000000000000000000000000000053200001

Initializes a new remote builder session

function

newBuilder ( )

internal

view

returns

(string

memory) Outputs:

- sessionid
- (string): ID of the remote builder session

newDataRecord

Address:0x0000000000000000000000000000000042030000

Initializes data records within the ConfidentialStore. Prior to storing data, all data records should undergo initialization via

this precompile.

function

newDataRecord (uint64 decryptionCondition ,

address [ ]

memory allowedPeekers ,

address [ ]

memory allowedStores ,

string

memory dataType)

internal

view

returns

(DataRecord memory) Inputs:

- decryptionCondition
- (uint64): Up to which block this data record is valid. Used duringfillMevShareBundle
- precompie.
- allowedPeekers
- (address[]): Addresses which can get data
- allowedStores
- (address[]): Addresses can set data
- dataType
- (string): Namespace of the data

Outputs:

- dataRecord
- (DataRecord): Data record that was created

privateKeyGen

Address:0x0000000000000000000000000000000053200003

Generates a private key in ECDA secp256k1 format

function

privateKeyGen (CryptoSignature crypto)

internal

view

returns

(string

memory) Inputs:

- crypto
- (CryptoSignature): Type of the private key to generate

Outputs:

- privateKey
- (string): Hex encoded string of the ECDSA private key. Exactly as a signMessage precompile wants.

randomBytes

Address:0x000000000000000000000000000000007770000b

Generates a number of random bytes, given by the argument numBytes.

function

randomBytes (uint8 numBytes)

internal

view

returns

(bytes

memory) Inputs:

- numBytes
- (uint8): Number of random bytes to generate

Outputs:

- value
- (bytes): Randomly-generated bytes

signEthTransaction

Address:0x0000000000000000000000000000000040100001

Signs an Ethereum Transaction, 1559 or Legacy, and returns raw signed transaction bytes.txn is binary encoding of the transaction.

function

signEthTransaction (bytes

memory txn ,

string

memory chainId ,

string

memory signingKey)

internal

view

returns

(bytes

memory) Inputs:

- txn
- (bytes): Transaction to sign (RLP encoded)
- chainId
- (string): Id of the chain to sign for (hex encoded, with 0x prefix)
- signingKey
- (string): Hex encoded string of the ECDSA private key (without 0x prefix)

Outputs:

- signedTxn
- (bytes): Signed transaction encoded in RLP

signMessage

Address:0x0000000000000000000000000000000040100003

Signs a message and returns the signature.

function

signMessage (bytes

memory digest , CryptoSignature crypto ,

string

memory signingKey)

internal

view

returns

(bytes

memory) Inputs:

- digest
- (bytes): Message to sign
- crypto
- (CryptoSignature): Type of the private key to generate
- signingKey
- (string): Hex encoded string of the ECDSA private key

Outputs:

- signature
- (bytes): Signature of the message with the private key

simulateBundle

Address:0x0000000000000000000000000000000042100000

Performs a simulation of the bundle by building a block that includes it.

function

simulateBundle (bytes

memory bundleData)

internal

view

returns

(uint64) Inputs:

- bundleData
- (bytes): Bundle encoded in JSON

Outputs:

- effectiveGasPrice
- (uint64): Effective Gas Price of the resultant block

simulateTransaction

Address:0x0000000000000000000000000000000053200002

Simulates a transaction on a remote builder session

function

simulateTransaction (string

memory sessionid ,

bytes

memory txn)

internal

view

returns

(SimulateTransactionResult memory) Inputs:

- sessionid
- (string): ID of the remote builder session
- txn
- (bytes): Txn to simulate encoded in RLP

Outputs:

- simulationResult
- (SimulateTransactionResult): Result of the simulation

submitBundleJsonRPC

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) Inputs:

- url
- (string): URL to send the request to
- method
- (string): JSONRPC method to call
- params

- (bytes): JSONRPC input params encoded in RLP

Outputs:

- errorMessage
- (bytes): Error message if any

submitEthBlockToRelay

Address:0x0000000000000000000000000000000042100002

Submits a given builderBid to a mev-boost relay.

function

submitEthBlockToRelay (string

memory relayUrl ,

bytes

memory builderBid)

internal

view

returns

(bytes

memory) Inputs:

- relayUrl
- (string): URL of the relay to submit to
- builderBid
- (bytes): Block bid to submit encoded in JSON

Outputs:

- blockBid
- (bytes): Error message if any

Precompiles Governance

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

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

Edit this page
Previous MEVM Next Confidential Computation