Interface: PXE

Private eXecution Environment (PXE) runs locally for each user, providing functionality for all the operations needed to interact with the Aztec network, including account management, private data management, transaction local simulation, and access to an Aztec node. This interface, as part of a Wallet, is exposed to dapps for interacting with the network on behalf of the user.

Methods

addAuthWitness

▸addAuthWitness (authWitness):Promise \<void

Insert an auth witness for a given message hash. Auth witnesses are used to authorize actions on behalf of a user. For instance, a token transfer initiated by a different address may request authorization from the user to move their tokens. This authorization is granted by the user account contract by verifying an auth witness requested to the execution oracle. Witnesses are usually a signature over a hash of the action to be authorized, but their actual contents depend on the account contract that consumes them.

Parameters

Name Type Description authWitness AuthWitness The auth witness to insert. Composed of an identifier, which is the hash of the action to be authorized, and the actual witness as an array of fields, which are to be deserialized and processed by the account contract.

Returns

Promise \<void

addCapsule

▸addCapsule (capsule):Promise \<void

Adding a capsule to the capsule dispenser.

Parameters

Name Type Description capsule Fr [] An array of field elements representing the capsule.

Returns

Promise \<void

Remarks

A capsule is a "blob" of data that is passed to the contract through an oracle.

addContracts

▸addContracts (contracts):Promise \<void

Adds deployed contracts to the PXE Service. Deployed contract information is used to access the contract code when simulating local transactions. This is automatically called by aztec.js when deploying a contract. Dapps that wish to interact with contracts already deployed should register these contracts in their users' PXE Service through this method.

Parameters

Name Type Description contracts DeployedContract [] An array of DeployedContract objects containing contract ABI, address, and portal contract.

Returns

Promise \<void

addNote

▸addNote (note):Promise \<void

Adds a note to the database.

Parameters

Name Type Description note ExtendedNote The note to add.

Returns

Promise \<void

Throws

If the note hash of the note doesn't exist in the tree.

getBlock

▸getBlock (number):Promise \<undefined |L2Block

Get the given block.

Parameters

Name Type Description number number The block number being requested.

Returns

Promise \<undefined |L2Block

The blocks requested.

getBlockNumber

▸getBlockNumber ():Promise \<number

Fetches the current block number.

Returns

Promise \<number

The block number.

getContractClass

▸getContractClass (id):Promise \<undefined |ContractClassWithId

Returns a Contact Class given its identifier. TODO(@spalladino): The PXE actually holds artifacts and not classes, what should we return? Also, should the pxe query the node for contract public info, and merge it with its own definitions?

Parameters

Name Type Description id Fr Identifier of the class.

Returns

Promise \<undefined |ContractClassWithId

getContractData

▸getContractData (contractAddress):Promise \<undefined |ContractData

Gets the portal contract address on L1 for the given contract.

Parameters

Name Type Description contractAddress AztecAddress The contract's address.

Returns

Promise \<undefined |ContractData

The contract's portal address if found.

getContractInstance

▸getContractInstance (address):Promise \<undefined |ContractInstanceWithAddress

Returns a Contact Instance given its address, which includes the contract class identifier, portal address, initialization hash, deployment salt, and public keys hash. TODO(@spalladino): Should we return the public keys in plain as well here?

Parameters

Name Type Description address AztecAddress Deployment address of the contract.

Returns

Promise \<undefined |ContractInstanceWithAddress

getContracts

▸getContracts ():Promise \

Retrieves the addresses of contracts added to this PXE Service.

Returns

Promise \

An array of contracts addresses registered on this PXE Service.

getExtendedContractData

▸getExtendedContractData (contractAddress):Promise \<undefined |ExtendedContractData

Gets the extended contract data for this contract. Extended contract data includes the address, portal contract address on L1, public functions, partial address, and encryption public key.

Parameters

Name Type Description contractAddress AztecAddress The contract's address.

Returns

Promise \<undefined |ExtendedContractData

The extended contract data if found.

getNodeInfo

▸getNodeInfo ():Promise \<NodeInfo

Returns the information about the server's node. Includes current Node version, compatible Noir version, L1 chain identifier, protocol version, and L1 address of the rollup contract.

Returns

Promise \<NodeInfo

- The node information.

getNoteNonces

▸getNoteNonces (note):Promise \

Finds the nonce(s) for a given note.

Parameters

Name Type Description note ExtendedNote The note to find the nonces for.

Returns

Promise \

The nonces of the note.

Remarks

More than a single nonce may be returned since there might be more than one nonce for a given note. TODO(#4956): Un-expose this

getNotes

▸getNotes (filter):Promise \

Gets notes of accounts registered in this PXE based on the provided filter.

Parameters

Name Type Description filter NoteFilter The filter to apply to the notes.

Returns

Promise \

The requested notes.

getPublicStorageAt

▸getPublicStorageAt (contract ,slot):Promise \<Fr

Gets the storage value at the given contract storage slot.

Parameters

Name Type Description contract AztecAddress Address of the contract to query. slot Fr Slot to query.

Returns

Promise \<Fr

Storage value at the given contract slot.

Remarks

The storage slot here refers to the slot as it is defined in Noir not the index in the merkle tree. Aztec's version ofeth_getStorageAt .

Throws

If the contract is not deployed.

getRecipient

▸getRecipient (address):Promise \<undefined |CompleteAddress

Retrieves the complete address of the recipient corresponding to the provided aztec address. Complete addresses include the address, the partial address, and the encryption public key.

Parameters

Name Type Description address AztecAddress The aztec address of the recipient.

Returns

Promise \<undefined |CompleteAddress

The complete address of the requested recipient.

getRecipients

▸getRecipients ():Promise \

Retrieves the recipients added to this PXE Service.

Returns

Promise \

An array of recipients registered on this PXE Service.

getRegisteredAccount

▸getRegisteredAccount (address):Promise \<undefined |CompleteAddress

Retrieves the complete address of the account corresponding to the provided aztec address. Complete addresses include the address, the partial address, and the encryption public key.

Parameters

Name Type Description address AztecAddress The address of account.

Returns

Promise \<undefined |CompleteAddress

The complete address of the requested account if found.

getRegisteredAccounts

▸getRegisteredAccounts ():Promise \

Retrieves the user accounts registered on this PXE Service.

Returns

Promise \

An array of the accounts registered on this PXE Service.

getSyncStatus

▸getSyncStatus ():Promise \<SyncStatus

Returns the latest block that has been synchronized globally and for each account. The global block number indicates whether global state has been updated up to that block, whereas each address indicates up to which block the private state has been synced for that account.

Returns

Promise \<SyncStatus

The latest block synchronized for blocks, and the latest block synched for notes for each public key being tracked.

getTxEffect

▸getTxEffect (txHash):Promise \<undefined |TxEffect

Get a tx effect.

Parameters

Name Type Description txHash TxHash The hash of a transaction which resulted in the returned tx effect.

Returns

Promise \<undefined |TxEffect

The requested tx effect.

getTxReceipt

▸getTxReceipt (txHash):Promise \<TxReceipt

Fetches a transaction receipt for a given transaction hash. Returns a mined receipt if it was added to the chain, a pending receipt if it's still in the mempool of the connected Aztec node, or a dropped receipt if not found in the connected Aztec node.

Parameters

Name Type Description txHash TxHash The transaction hash.

Returns

Promise \<TxReceipt

A receipt of the transaction.

getUnencryptedLogs

▸getUnencryptedLogs (filter):Promise \<GetUnencryptedLogsResponse

Gets unencrypted logs based on the provided filter.

Parameters

Name Type Description filter LogFilter The filter to apply to the logs.

Returns

Promise \<GetUnencryptedLogsResponse

The requested logs.

isAccountStateSynchronized

▸isAccountStateSynchronized (account):Promise \<boolean

Checks if the specified account is synchronized.

Parameters

Name Type Description account AztecAddress The aztec address for which to query the sync status.

Returns

Promise \<boolean

True if the account is fully synched, false otherwise.

Deprecated

UsegetSyncStatus instead.

Remarks

Checks whether all the notes from all the blocks have been processed. If it is not the case, the retrieved information from contracts might be old/stale (e.g. old token balance).

Throws

If checking a sync status of account which is not registered.

isContractClassPubliclyRegistered

▸isContractClassPubliclyRegistered (id):Promise \<boolean

Queries the node to check whether the contract class with the given id has been publicly registered. TODO(@spalladino): This method is strictly needed to decide whether to publicly register a class or not during a public deployment. We probably want a nicer and more general API for this, but it'll have to do for the time being.

Parameters

Name Type Description id Fr Identifier of the class.

Returns

Promise \<boolean

isGlobalStateSynchronized

▸isGlobalStateSynchronized ():Promise \<boolean

Checks whether all the blocks were processed (tree roots updated, txs updated with block info, etc.).

Returns

Promise \<boolean

True if there are no outstanding blocks to be synched.

Remarks

This indicates that blocks and transactions are synched even if notes are not. Compares local block number with the block number from aztec node.

Deprecated

UsegetSyncStatus instead.

registerAccount

▸registerAccount (privKey ,partialAddress):Promise \<CompleteAddress

Registers a user account in PXE given its master encryption private key. Once a new account is registered, the PXE Service will trial-decrypt all published notes on the chain and store those that correspond to the registered account. Will do nothing if the account is already registered.

Parameters

Name Type Description privKey Fq Private key of the corresponding user master public key. partialAddress Fr The partial address of the account contract corresponding to the account being registered.

Returns

Promise \<CompleteAddress

The complete address of the account.

registerRecipient

▸registerRecipient (recipient):Promise \<void

Registers a recipient in PXE. This is required when sending encrypted notes to a user who hasn't deployed their account contract yet. Since their account is not deployed, their encryption public key has not been broadcasted, so we need to manually register it on the PXE Service in order to be able to encrypt data for this recipient.

Parameters

Name Type Description recipient CompleteAddress The complete address of the recipient

Returns

Promise \<void

Remarks

Called recipient because we can only send notes to this account and not receive them via this PXE Service. This is because we don't have the associated private key and for this reason we can't decrypt the recipient's notes. We can send notes to this account because we can encrypt them with the recipient's public key.

sendTx

▸sendTx (tx):Promise \<TxHash

Sends a transaction to an Aztec node to be broadcasted to the network and mined.

Parameters

Name Type Description tx Tx The transaction as created viasimulateTx .

Returns

Promise \<TxHash

A hash of the transaction, used to identify it.

simulateTx

▸simulateTx (txRequest ,simulatePublic):Promise \<Tx

Creates a transaction based on the provided preauthenticated execution request. This will run a local simulation of the private execution (and optionally of public as well), assemble the zero-knowledge proof for the private execution, and return the transaction object.

Parameters

Name Type Description txRequest TxExecutionRequest An authenticated tx request ready for simulation simulatePublic boolean Whether to simulate the public part of the transaction.

Returns

Promise \<Tx

A transaction ready to be sent to the network for execution.

Throws

If the code for the functions executed in this transaction has not been made available viaaddContracts . Also throws if simulatePublic is true and public simulation reverts.

viewTx

▸viewTx (functionName ,args ,to ,from?):Promise \<any

Simulate the execution of a view (read-only) function on a deployed contract without actually modifying state. This is useful to inspect contract state, for example fetching a variable value or calling a getter function. The function takes function name and arguments as parameters, along with the contract address and optionally the sender's address.

Parameters

Name Type Description functionName string The name of the function to be called in the contract. args any [] The arguments to be provided to the function. to AztecAddress The address of the contract to be called. from? AztecAddress (Optional) The msg sender to set for the call.

Returns

Promise \<any

The result of the view function call, structured based on the function ABI.

Previous Cheat Codes Next Smart Contracts