Taming Monero: The Hidden Model

A Specification for

Monero Wallet and Daemon Interfaces and Types

based on Monero Core v0.14.1.0 Boron Butterfly

Revision 25 - August 25, 2019

Java reference implementation: https://github.com/monero-ecosystem/monero-java

JavaScript reference implementation: https://github.com/monero-ecosystem/monero-javascript

C++ reference implementation: https://github.com/woodser/monero-cpp-library

Document source: https://github.com/monero-ecosystem/monero-java/blob/master/monero-spec.xml

Document source created and modifiable using: https://www.draw.io/

Sample Daemon and Wallet Code

This code demonstrates how to use the Java library with a native binding to Monero Core. All libraries conform to the interfaces and types specified in this document.

```
// connect to a daemon
MoneroDaemon daemon = new MoneroDaemonRpc("http://localhost:38081");
long height = daemon.getHeight();                 // 1523651
BigInteger feeEstimate = daemon.getFeeEstimate(); // 1014313512

// get transactions in the pool
List<MoneroTx> txsInPool = daemon.getTxPool();
for (MoneroTx tx : txsInPool) {
  String id = tx.getId();
  BigInteger fee = tx.getFee();
  boolean isDoubleSpendSeen = tx.isDoubleSpendSeen();
}

// get last 100 blocks as a binary request
List<MoneroBlock> blocks = daemon.getBlocksByRange(height - 100, height - 1);
for (MoneroBlock block : blocks) {
  int numTxs = block.getTxs().size();
}

// connect to a wallet using RPC
MoneroWalletRpc walletRPC = new MoneroWalletRpc("http://localhost:38083", "rpc_user", "abc123");
String primaryAddress = walletRPC.getPrimaryAddress(); // 59aZULsUF3YNSKGiHz4J...
BigInteger balance = walletRPC.getBalance();           // 533648366742
MoneroSubaddress subaddress = walletRPC.getSubaddress(1, 0);
BigInteger subaddressBalance = subaddress.getBalance();

// query a transaction by id
MoneroTxWallet tx = walletRPC.getTx("314a0f1375db31cea4dac4e0a51514a6282b43792269b3660166d4d2b46437ca");
long txHeight = tx.getHeight();
List<MoneroIncomingTransfer> incomingTransfers = tx.getIncomingTransfers();
List<MoneroDestination> destinations = tx.getOutgoingTransfer().getDestinations();

// query incoming transfers to account 1
MoneroTransferQuery transferQuery = new MoneroTransferQuery().setIsIncoming(true).setAccountIndex(1);
List<MoneroTransfer> transfers = walletRPC.getTransfers(transferQuery);

// query unspent outputs
MoneroOutputQuery outputQuery = new MoneroOutputQuery().setIsSpent(false);
List<MoneroOutputWallet> outputs = walletRPC.getOutputs(outputQuery);

// create a wallet from a mnemonic phrase using Java native bindings to Monero Core
MoneroWalletJni walletJNI = MoneroWalletJni.createWalletFromMnemonic("MyWallet",
    "supersecretpassword123", MoneroNetworkType.STAGENET, "hefty value ...",
    new MoneroRpcConnection("http://localhost:38081"), 384151l);

// synchronize the wallet and receive progress notifications
walletJNI.sync(new MoneroSyncListener() {
  @Override
  public void onSyncProgress(long height, long startHeight, long endHeight, double percentDone,
      String message) {
    // feed a progress bar?
  }
});

// start syncing the wallet continuously in the background
walletJNI.startSyncing();
```

```
// be notified when the JNI wallet receives funds
walletJNI.addListener(new MoneroWalletListener() {

  @Override
  public void onOutputReceived(MoneroOutputWallet output) {
    System.out.println("Wallet received funds!");
    String txId = output.getTx().getId();
    int accountIdx = output.getAccountIndex();
    int subaddressIdx = output.getSubaddressIndex();
    JNI_OUTPUT_RECEIVED = true;
  }
});

// send funds from the RPC wallet to the JNI wallet
MoneroTxWallet sentTx = walletRPC.send(0, walletJNI.getPrimaryAddress(), new BigInteger("50000"));
assertTrue(sentTx.inTxPool());

// mine with 7 threads to push the network along
int numThreads = 7;
boolean isBackground = false;
boolean ignoreBattery = false;
walletRPC.startMining(numThreads, isBackground, ignoreBattery);

// wait for the next block to be added to the chain
MoneroBlockHeader nextBlockHeader = daemon.getNextBlockHeader();
long nextNumTxs = nextBlockHeader.getNumTxs();

// stop mining
walletRPC.stopMining();

// the transaction is (probably) confirmed
TimeUnit.SECONDS.sleep(10); // let the wallet refresh
boolean isConfirmed = walletRPC.getTx(sentTx.getId()).isConfirmed();

// create a request to send funds from the RPC wallet to multiple destinations in the JNI wallet
MoneroSendRequest request = new MoneroSendRequest()
        .setAccountIndex(1)                           // send from account 1
        .setSubaddressIndices(0, 1)                   // send from subaddreses in account 1
        .setPriority(MoneroSendPriority.UNIMPORTANT)  // no rush
        .setDestinations(
                new MoneroDestination(walletJNI.getAddress(1, 0), new BigInteger("50000")),
                new MoneroDestination(walletJNI.getAddress(2, 0), new BigInteger("50000")));

// create the transaction, confirm with the user, and relay to the network
MoneroTxWallet createdTx = walletRPC.createTx(request);
BigInteger fee = createdTx.getFee();  // "Are you sure you want to send ...?"
walletRPC.relayTx(createdTx); // submit the transaction which will notify the JNI wallet

// JNI wallet will receive notification of incoming output after a moment
TimeUnit.SECONDS.sleep(10);
assertTrue(JNI_OUTPUT_RECEIVED);

// save and close the JNI wallet
walletJNI.close(true);
```

Sample Multisig Wallet Creation

This code demonstrates a utility for creating N/N, (N-1)/N and M/N multisig wallets using this library.

```
public static List<MoneroWalletJni> createMultisigParticipants(int M, int N) {

  // create participating wallets
  List<MoneroWalletJni> participants = new ArrayList<MoneroWalletJni>();
  for (int i = 0; i < N; i++) {
    MoneroWalletJni participant = MoneroWalletJni.createWalletFromMnemonic("MyWallet",
        "abc123", MoneroNetworkType.STAGENET, "hefty value ...",
        new MoneroRpcConnection("http://localhost:38081"), 384151l);
    participants.add(participant);
  }

  // prepare and collect multisig hex from each participant
  List<String> preparedMultisigHexes = new ArrayList<String>();
  for (MoneroWallet participant : participants) preparedMultisigHexes.add(participant.prepareMultisig());

  // make each wallet multsig and collect results
  List<String> madeMultisigHexes = new ArrayList<String>();
  for (int i = 0; i < participants.size(); i++) {

    // collect prepared multisig hexes from wallet's peers
    List<String> peerMultisigHexes = new ArrayList<String>();
    for (int j = 0; j < participants.size(); j++) if (j != i) {
      peerMultisigHexes.add(preparedMultisigHexes.get(j));
    }

    // make wallet multisig and collect result hex
    MoneroMultisigInitResult result = participants.get(i).makeMultisig(peerMultisigHexes, M, "abc123");
    madeMultisigHexes.add(result.getMultisigHex());
  }

  // if wallet is (N-1)/N, finalize each participant
  if (M == N - 1) {
    for (int i = 0; i < participants.size(); i++) {

      // collect made multisig hexes from wallet's peers
      List<String> peerMultisigHexes = new ArrayList<String>();
      for (int j = 0; j < participants.size(); j++)
          if (j != i) peerMultisigHexes.add(madeMultisigHexes.get(j));

      // finalize the multisig wallet using peer hex
      String primaryAddress = participants.get(i).finalizeMultisig(peerMultisigHexes,
          TestUtils.WALLET_PASSWORD);
    }
  }

  // if wallet is M/N, exchange multisig keys N-M times
  else if (M != N) {
    List<String> multisigHexes = madeMultisigHexes;
    for (int i = 0; i < N - M; i++) {

      // exchange multisig keys among participants and collect results for next round if applicable
      List<String> resultMultisigHexes = new ArrayList<String>();
      for (MoneroWallet participant : participants) {

        // import the multisig hex of other participants and collect results
        MoneroMultisigInitResult result = participant.exchangeMultisigKeys(multisigHexes, "abc123");
        resultMultisigHexes.add(result.getMultisigHex());
      }

      // use resulting multisig hex for next round of exchange if applicable
      multisigHexes = resultMultisigHexes;
    }
  }

  return participants;  // return participant wallets
}
```

Monero Daemon API 1/2

MoneroDaemon
isTrusted(): boolean
getHeight(): uint
getBlockId(height): string
getBlockTemplate(walletAddress, reserveSize): MoneroBlockTemplate
getLastBlockHeader(): MoneroBlockHeader
getBlockHeaderById(blockId): MoneroBlockHeader
getBlockHeaderByHeight(height): MoneroBlockHeader
getBlockHeadersByRange(startHeight, endHeight): list<MoneroBlockHeader>
getBlockById(blockId): MoneroBlock
getBlocksById(blockIds, startHeight, prune): list<MoneroBlock>
getBlockByHeight(height): MoneroBlock
getBlocksByHeight(heights): list<MoneroBlock>
getBlocksByRange(startHeight, endHeight): list<MoneroBlock>
getBlocksByRangeChunked(startHeight, endHeight, maxChunkSize): list<MoneroBlock>
getBlockIds(blockIds, startHeight): list<string>
getTx(txId, prune): MoneroTx
getTxs(txIds, prune): list<MoneroTx>
getTxHex(txId, prune): string
getTxHexes(txIds, prune): list<string>
getCoinbaseTxSum(height, numBlocks): MoneroCoinbaseTxSum
getFeeEstimate(graceBlocks): BigInteger
submitTxHex(txHex, doNotRelay): MoneroSubmitTxResult
relayTxById(txId): string
relayTxsById(txIds): list<string>
getTxPool(): list<MoneroTx>
getTxPoolIds(): list<string>
getTxPoolBacklog(): list<MoneroTxBacklogEntry>
getTxPoolStats(): MoneroTxPoolStats
flushTxPool(ids): list<string>
getSpentStatus(keyImage): MoneroKeyImageSpentStatus
getSpentStatuses(keyImages): list<MoneroKeyImageSpentStatus>
getOutputs(outputs): list<MoneroOutput>
getOutputHistogram(amounts, minCount, maxCount, isUnlocked, recentCutoff): list<MoneroOutputHistogramEntry>
getOutputDistribution(amounts, cumulative, startHeight, endHeight): list<MoneroOutputDistributionEntry>
getInfo(): MoneroDaemonInfo
getSyncInfo(): MoneroDaemonSyncInfo
getHardForkInfo(): MoneroDaemonHardForkInfo
getAltChains(): list<MoneroAltChain>

Monero Daemon API 2/2

getAltBlockIds(): list<string>

getDownloadLimit(): uint

setDownloadLimit(limit): uint

resetDownloadLimit(): uint

getUploadLimit(): uint

setUploadLimit(limit): uint

resetUploadLimit(): uint

getKnownPeers(): list<MoneroDaemonPeer>

getConnections(): list<MoneroDaemonConnection>

setOutgoingPeerLimit(limit): void

setIncomingPeerLimit(limit): void

getPeerBans(): list<MoneroBan>

setPeerBan(ban): void

setPeerBans(bans): void

startMining(address, numThreads, isBackground, ignoreBattery): void

stopMining(): void

getMiningStatus(): MoneroMiningStatus

submitBlock(blockBlob): void

submitBlocks(blockBlobs): void

checkForUpdate(): MoneroDaemonUpdateCheckResult

downloadUpdate(path): MoneroDaemonDownloadUpdateResult

stop(): void

Monero Daemon Types 1/3

MoneroBlockHeader
id: string
height: uint
timestamp: long
size: long
weight: long
long_term_weight: uint
depth: long
difficulty: BigInteger
cumulative_difficulty: BigInteger
major_version: uint
minor_version: uint
nonce: long
miner_tx_id: string
num_txs: uint
orphan_status: boolean
prev_id: string
reward: BigInteger
pow_hash: string

Extends

MoneroBlock
hex: string
miner_tx: MoneroTx
txs: list<MoneroTx>
tx_ids: list<string>

MoneroOutput
tx: MoneroTx
key_image: MoneroKeyImage
amount: BigInteger
index: uint
ring_output_indices: list<uint>
stealth_public_key: string

MoneroDaemonInfo
version: string
num_alt_blocks: uint
block_size_limit: uint
block_size_median: uint
block_weight_limit: uint
block_weight_median: uint
bootstrap_daemon_address: string
difficulty: BigInteger
cumulative_difficulty: uint
free_space: uint
num_offline_peers: uint
num_online_peers: uint
height: uint
height_without_bootstrap: uint
network_type: MoneroNetworkType
is_offline: boolean
num_incoming_connections: uint
num_outgoing_connections: uint
num_rpc_connections: uint
start_timestamp: uint
target: uint
target_height: uint
top_block_id: string
num_txs: uint
num_txs_pool: uint
was_bootstrap_ever_used: boolean
database_size: uint
update_available: boolean

MoneroTx
block: MoneroBlock
height: uint
id: string
version: uint
is_coinbase: boolean
payment_id: string
fee: BigInteger
mixin: uint
do_not_relay: boolean
is_relayed: boolean
is_confirmed: boolean
in_tx_pool: boolean
num_confirmations: uint
unlock_time: uint
last_relayed_timestamp: uint
received_timestamp: uint
is_double_spend: boolean
key: string
full_hex: string
pruned_hex: string
prunable_hex: string
prunable_hash: string
size: uint
weight: uint
vins: list<MoneroOutput>
vouts: list<MoneroOutput>
output_indices: list<uint>
metadata: string
extra: list<uint>
rct_signatures: list<string>
rct_sig_prunable: TODO
is_kept_by_block: boolean
is_failed: boolean
last_failed_height: uint
last_failed_id: string
max_used_block_height: uint

Monero Daemon Types 2/3

MoneroDaemonSyncInfo
height: uint
connections: list<MoneroDaemonConnection>
spans: list<MoneroDaemonConnectionSpan>
target_height: uint
next_needed_pruning_seed: uint
overview: ?

MoneroDaemonConnection
peer: MoneroDaemonPeer
id: string
avg_download: uint
avg_upload: uint
current_download: uint
current_upload: uint
height: uint
is_incoming: boolean
live_time: long
is_local_ip: boolean
is_local_host: boolean
num_receives: uint
num_sends: uint
receive_idle_time: long
send_idle_time: long
state: string
num_support_flags: uint

MoneroDaemonConnectionSpan
connection_id: string
num_blocks: uint
remote_address: string
rate: uint
speed: uint
size: uint
start_block_height: BigInteger

<<enumeration>> MoneroNetworkType
mainnet: 0
testnet: 1
stagenet: 2

MoneroKeyImage
hex: string
signature: string

<<enumeration>> MoneroKeyImageSpentStatus
not_spent: 0
confirmed: 1
tx_pool: 2

MoneroSubmitTxResult
is_good: boolean
is_relayed: boolean
is_double_spend_seen: boolean
is_fee_too_low: boolean
is_mixin_too_low: boolean
has_invalid_input: boolean
has_invalid_output: boolean
is_rct: boolean
is_overspend: boolean
is_too_big: boolean
sanity_check_failed
reason: string

	max_used_block_id: string
	signatures: list<string>

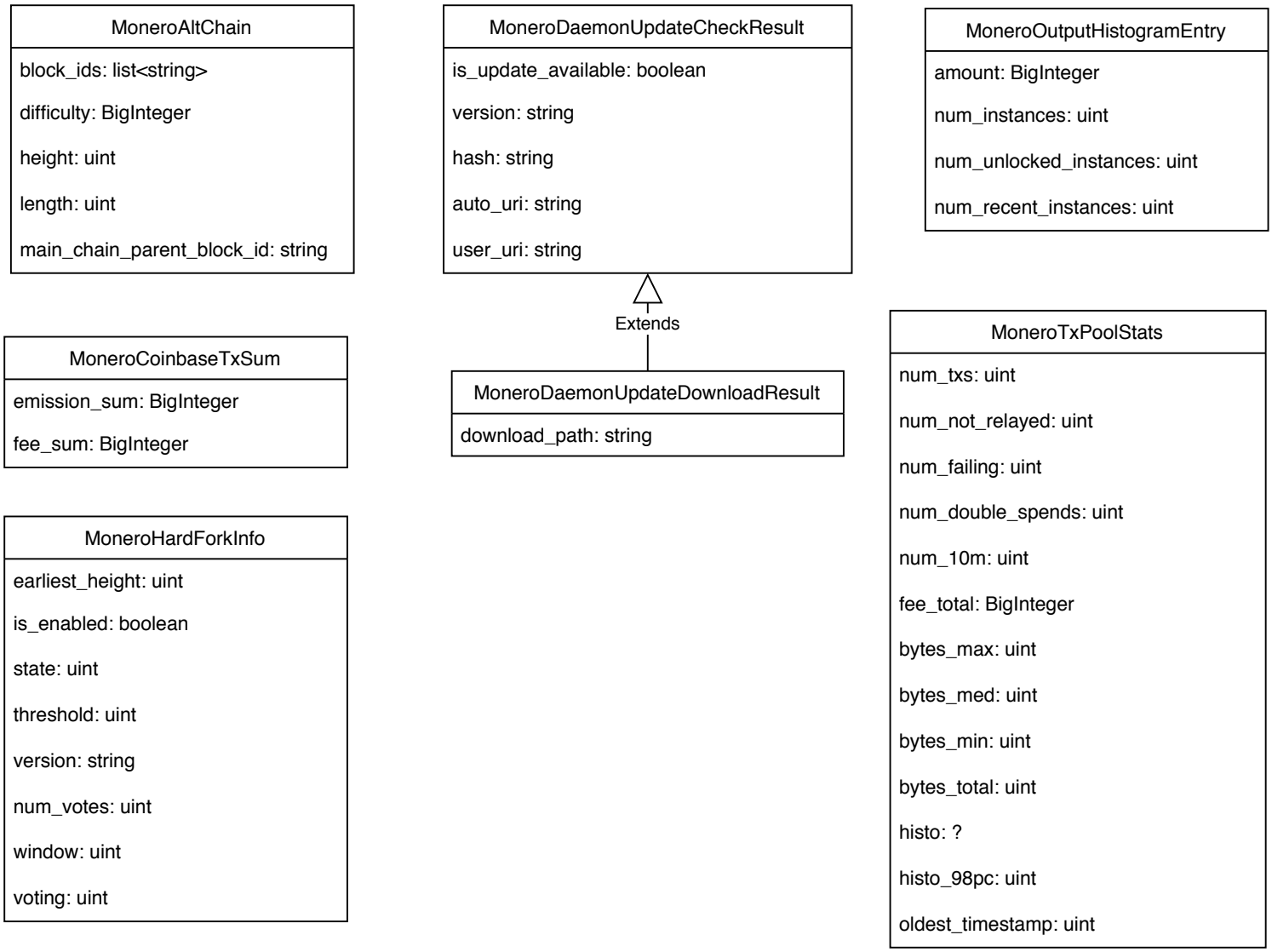
MoneroDaemonPeer
id: string
address: string
host: string
port: uint
rpc_port: uint
is_online: boolean
last_seen_timestamp: uint
pruning_seed: uint

MoneroBan
host: string
ip: string
is_banned: boolean
seconds: long

MoneroBlockTemplate
block_template_blob: string
block_hashing_blob: string
difficulty: BigInteger
expected_reward: BigInteger
height: uint
prev_id: string
reserved_offset: uint

MoneroMiningStatus
is_active: boolean
is_background: boolean
address: string
speed: uint
num_threads: uint

Monero Daemon Types 3/3



Monero Wallet API 1/2

MoneroWallet

getNetworkType(): MoneroNetworkType

getDaemonConnection(): MoneroRpcConnection

getSeed(): string

getMnemonic(): string

getLanguage(): list<string>

getLanguages(): list<string>

getPublicViewKey(): string

getPrivateViewKey(): string

getPublicSpendKey(): string

getPrivateSpendKey(): string

getPrimaryAddress(): string

getHeight(): uint

getDaemonHeight(): uint

getDaemonMaxPeerHeight(): uint

getApproximateChainHeight(): uint

getIntegratedAddress(paymentId): MoneroIntegratedAddress

decodeIntegratedAddress(integratedAddress): MoneroIntegratedAddress

sync(startHeight, onProgress): MoneroSyncResult

startSyncing()

stopSyncing()

isMultisigImportNeeded(): boolean

getAccounts(includeSubaddresses): list<MoneroAccount>

getAccount(accountIdx, includeSubaddresses): MoneroAccount

createAccount(label): MoneroAccount

getSubaddresses(accountIdx, subaddressIndices): list<MoneroSubaddress>

getSubaddress(accountIdx, subaddressIdx): MoneroSubaddress

createSubaddress(accountIdx, label): MoneroSubaddress

getAddress(accountIdx, subaddressIdx): string

getAddressIndex(address): uint

getBalance(accountIdx, subaddressIdx): BigInteger

getUnlockedBalance(accountIdx, subaddressIdx): BigInteger

getTxs(txQuery): list<MoneroTxWallet>

getTransfers(transferQuery): list<MoneroTransfer>

getIncomingTransfers(transferQuery): list<MoneroIncomingTransfer>

getOutgoingTransfers(transferQuery): list<MoneroOutgoingTransfer>

getOutputs(outputQuery): list<MoneroOutputWallet>

getOutputsHex(): string

importOutputsHex(outputsHex): uint

getKeyImages(): list<MoneroKeyImage>

MoneroSendRequest

destinations: list<MoneroDestination>

payment_id: string

priority: MoneroSendPriority

mixin: uint

ring_size: uint

fee: BigInteger

account_index: uint

subaddress_indices: list<uint>

unlock_time: uint

can_split: boolean

do_not_relay: boolean

note: string

recipient_name: string

below_amount: BigInteger

sweep_each_subaddress: boolean

key_image: string

Configures outgoing transfers, sweeps, and creation of payment URIs.

MoneroTxQuery extends MoneroTxWallet

is_outgoing: bool

is_incoming: bool

tx_ids: list<string>

has_payment_id: bool

payment_ids: list<string>

height: uint

min_height: uint

max_height: uint

include_outputs: bool

transfer_request: MoneroTransferRequest

output_request: MoneroOutputRequest

Configures a query to get wallet transactions, allowing filtering on all transaction fields and extensions.

importKeyImages(keyImages): MoneroKeyImageImportResult

getNewKeyImagesFromLastImport(): list<MoneroKeyImage>

createTx(sendRequest): MoneroTxSet

createTxs(sendRequest): MoneroTxSet

relayTx(txMetadata): string

relayTxs(txMetadatas): list<string>

send(sendRequest): MoneroTxSet

sendSplit(sendRequest): MoneroTxSet

sweepSubaddress(accountIdx, subaddressIdx, address): MoneroTxSet

sweepAccount(accountIdx, address): MoneroTxSet

sweepWallet(address): list<MoneroTxSet>

sweepUnlocked(sendRequest): list<MoneroTxSet>

sweepDust(doNotRelay): MoneroTxSet

sweepOutput(sendRequest): MoneroTxSet

getTxNote(txId): string

setTxNote(txId, note): void

getTxNotes(txIds): list<string>

setTxNotes(txIds, notes): void

sign(msg): string

verify(msg, address, signature): boolean

getTxKey(txId): string

checkTxKey(txId, txKey, address): MoneroCheckTx

getTxProof(txId, address, message): string

checkTxProof(txId, address, message, signature): MoneroCheckTx

getSpendProof(txId, message): string

checkSpendProof(txId, message, signature): boolean

getReserveProofWallet(message): string

getReserveProofAccount(accountIdx, amount, message): string

checkReserveProof(address, message, signature): MoneroCheckReserve

getAddressBookEntries(entryIndices): list<MoneroAddressBookEntry>

addAddressBookEntry(address, description, paymentId): uint

deleteAddressBookEntry(entryIdx): void

getTxProof(txId, address, message): string

tagAccounts(tag, accountIndices): void

untagAccounts(accountIndices): void

getAccountTags(): list<MoneroAccountTag>

setAccountTagLabel(tag, label): void

createPaymentUri(sendRequest): string

parsePaymentUri(uri): MoneroSendRequest

setAttribute(key, val): void

getAttribute(key): string

startMining(numThreads, backgroundMining, ignoreBattery): void

Monero Wallet API 2/2

MoneroTransferQuery extends MoneroTransfer
is_incoming: bool
address: string
addresses: list<string>
subaddress_index: uint
subaddress_indices: list<uint>
destinations: list<MoneroDestination>
has_destinations: bool
tx_request: MoneroTxRequest

Configures a query to get wallet transfers, allowing filtering on all transfer fields and extensions.

MoneroOutputQuery extends MoneroOutputWallet
subaddress_indices: list<uint>
tx_request: MoneroTxRequest

Configures a query to get wallet outputs, allowing filtering on all output fields and extensions.

stopMining(): void

moveTo(path, password): void

save(): void

close(bool save): void

Monero Wallet Types 1/1

MoneroAccount
index: uint
primary_address: string
balance: BigInteger
unlocked_balance: BigInteger
subaddresses: list<MoneroSubaddress>
tag: string

MoneroSubaddress
account_index: uint
index: uint
address: string
label: string
balance: BigInteger
unlocked_balance: BigInteger
is_used: boolean
num_unspent_outputs: uint
num_blocks_to_unlock: uint

<<enumeration>> MoneroSendPriority
default: 0
unimportant: 1
normal: 2
elevated: 3

MoneroAccountTag
tag: string
label: string
account_indices: list<int>

MoneroTxSet
txs: list<MoneroTxWallet>
multisig_tx_hex: string
unsigned_tx_hex: string
signed_tx_hex: string

MoneroAddressBookEntry
index: uint
address: string
payment_id: string
description: string

MoneroDestination
address: string
amount: BigInteger

MoneroTxWallet extends MoneroTx
tx_set: MoneroTxSet
incoming_amount: BigInteger
outgoing_amount: BigInteger
incoming_transfers: list<MoneroTransfer>
outgoing_transfer: MoneroTransfer
note: string

MoneroSyncResult
num_blocks_fetched: uint
received_money: boolean

MoneroTransfer
tx: MoneroTxWallet
is_incoming: boolean
amount: BigInteger
account_index: uint
num_suggested_confirmations: uint

MoneroKeyImageImportResult
height: uint
spent_amount: BigInteger
unspent_amount: BigInteger

MoneroOutputWallet extends MoneroOutput
account_index: uint
subaddress_index: uint
is_spent: boolean
is_unlocked: boolean
is_frozen: boolean

Extends

Extends

MoneroIncomingTransfer
subaddress_index: uint
address: string

MoneroOutgoingTransfer
subaddress_indices: list<uint>
addresses: list<string>
destinations: list<MoneroDestination>

MoneroCheck
is_good: boolean

Extends

Extends

MoneroIntegratedAddress (deprecated)
standard_address: string
payment_id: string
integrated_address: string

MoneroCheckReserve
total_amount: BigInteger
unconfirmed_spent_amount: BigInteger

MoneroCheckTx
in_tx_pool: boolean
num_confirmations: uint
received_amount: BigInteger

woodser API 1/1

46FR1GKVqFNQnDiFkH7AuzbUBrGQwz2VdaXTDD4jcjRE8YkkoTYTmZ2Vohsz9gLSqkj5EM6ai9Q7sBoX4FPPYJdGKQQXPVz