Taming Monero: The Hidden Model

A Specification for

Monero Wallet and Daemon Interfaces

based on Monero Core v0.14.1.0 Boron Butterfly

Revision 27 - September 04, 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<MoneroWallet> createMultisigParticipants(int M, int N) {

  // create participating wallets
  List<MoneroWallet> participants = new ArrayList<MoneroWallet>();
  for (int i = 0; i < N; i++) {
    MoneroWallet 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 not N/N, exchange multisig keys N-M times
  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 participant wallets
  return participants;
}
```

Monero Daemon Interface 1/2

isTrusted(): bool

getHeight(): ulong

getBlockId(ulong height): string

getBlockTemplate(string walletAddress, uint reserveSize=null): MoneroBlockTemplate

getLastBlockHeader(): MoneroBlockHeader

getBlockHeaderById(string blockId): MoneroBlockHeader

getBlockHeaderByHeight(ulong height): MoneroBlockHeader

getBlockHeadersByRange(ulong startHeight=0, ulong endHeight=chainHeight): MoneroBlockHeader[]

getBlockById(string blockId): MoneroBlock

getBlocksById(string[] blockIds, ulong startHeight, bool prune=false): MoneroBlock[]

getBlockByHeight(ulong height): MoneroBlock

getBlocksByHeight(ulong[] heights): MoneroBlock[]

getBlocksByRange(ulong startHeight=0, ulong endHeight=chainHeight): MoneroBlock[]

getBlocksByRangeChunked(ulong startHeight=0, ulong endHeight=chainHeight, ulong maxChunkSize=3000000): MoneroBlock[]

getBlockIds(string[] blockIds, ulong startHeight): string[]

getTx(string txId, bool prune=false): MoneroTx

getTxs(string[] txIds, bool prune=false): MoneroTx[]

getTxHex(string txId, bool prune=false): string

getTxHexes(string[] txIds, bool prune=false): string[]

getMinerTxSum(ulong height, ulong numBlocks=chainHeight): MoneroMinerTxSum

getFeeEstimate(ulong graceBlocks=?): ulong

submitTxHex(string txHex, bool doNotRelay=false): MoneroSubmitTxResult

relayTxById(string txId): void

relayTxsById(string[] txIds): void

getTxPool(): MoneroTx[]

getTxPoolIds(): string[]

getTxPoolBacklog(): MoneroTxBacklogEntry[]

getTxPoolStats(): MoneroTxPoolStats

flushTxPool(string[] txIds=null): void

getKeyImageSpentStatus(string keyImage): MoneroKeyImageSpentStatus

getKeyImageSpentStatuses(string[] keyImages): MoneroKeyImageSpentStatus[]

getOutputs(MoneroOutput[] outputs): MoneroOutput[]

getOutputHistogram(ulong[] amounts, ulong minCount=?, ulong maxCount=?, bool isUnlocked=null, ulong recentCutoff=?): MoneroOutputHistogramEntry[]

getOutputDistribution(ulong[] amounts, bool isCumulative=?, ulong startHeight=0, ulong endHeight=chainHeight): MoneroOutputDistributionEntry[]

getInfo(): MoneroDaemonInfo

getSyncInfo(): MoneroDaemonSyncInfo

getHardForkInfo(): MoneroHardForkInfo

Monero Daemon Intereface 2/2

getAltChains(): MoneroAltChain[]

getAltBlockIds(): string[]

getDownloadLimit(): uint

setDowloadLimit(uint limit): uint

resetDownloadLimit(): uint

getUploadLimit(): uint

setUploadLimit(uint): uint

resetUploadLimit(): uint

getKnownPeers(): MoneroDaemonPeer[]

getConnections(): MoneroDaemonConnection[]

setOutgoingPeerLimit(uint limit): void

setIncomingPeerLimit(uint limit): void

getPeerBans(): MoneroBan[]

setPeerBan(MoneroBan ban): void

setPeerBans(MoneroBan[] bans): void

startMining(string address, ulong numThreads=null, bool isBackground=false, bool ignoreBattery=false): void

stopMining(): void

getMiningStatus(): MoneroMiningStatus

submitBlock(string blockBlob): void

submitBlocks(string[] blockBlobs): void

checkForUpdate(): MoneroDaemonUpdateCheckResult

downloadUpdate(string path): MoneroDaemonUpdateDownloadResult

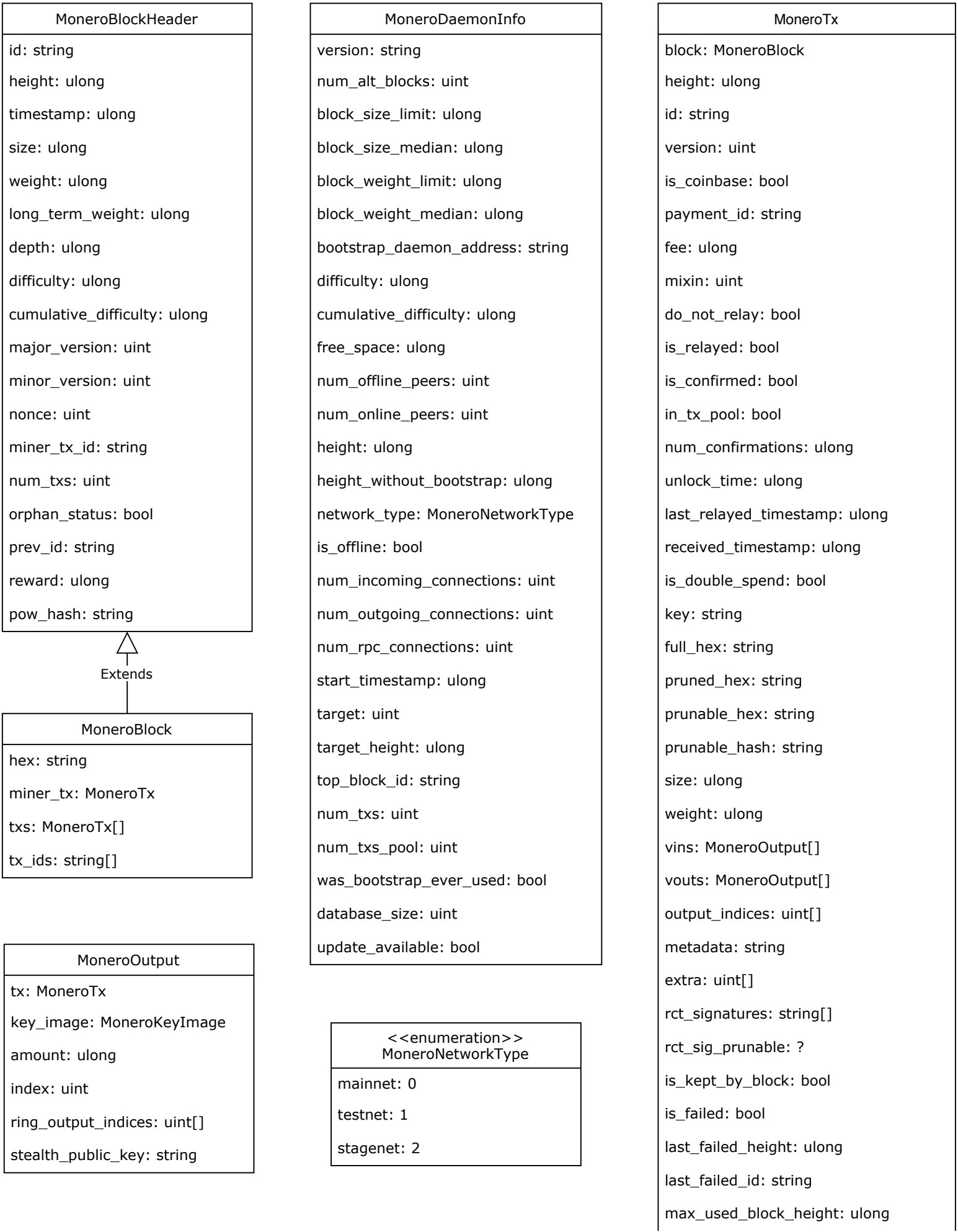
getNextBlockHeader(): MoneroBlockHeader

addListener(MoneroDaemonListener listener): void

removeListener(MoneroDaemonListener listener): void

stop(): void

Monero Daemon Types 1/3



Monero Daemon Types 2/3

MoneroDaemonSyncInfo
height: ulong
connections: MoneroDaemonConnection[]
spans: MoneroDaemonConnectionSpan[]
target_height: ulong
next_needed_pruning_seed: uint
overview: ?

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

MoneroDaemonConnectionSpan
connection_id: string
num_blocks: ulong
remote_address: string
rate: ulong
speed: ulong
size: ulong
start_block_height: ulong

MoneroDaemonListener
onBlockHeader(MoneroBlockHeader header): void

MoneroKeyImage
hex: string
signature: string

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

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

MoneroMinerTxSum
emission_sum: ulong
fee_sum: ulong

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

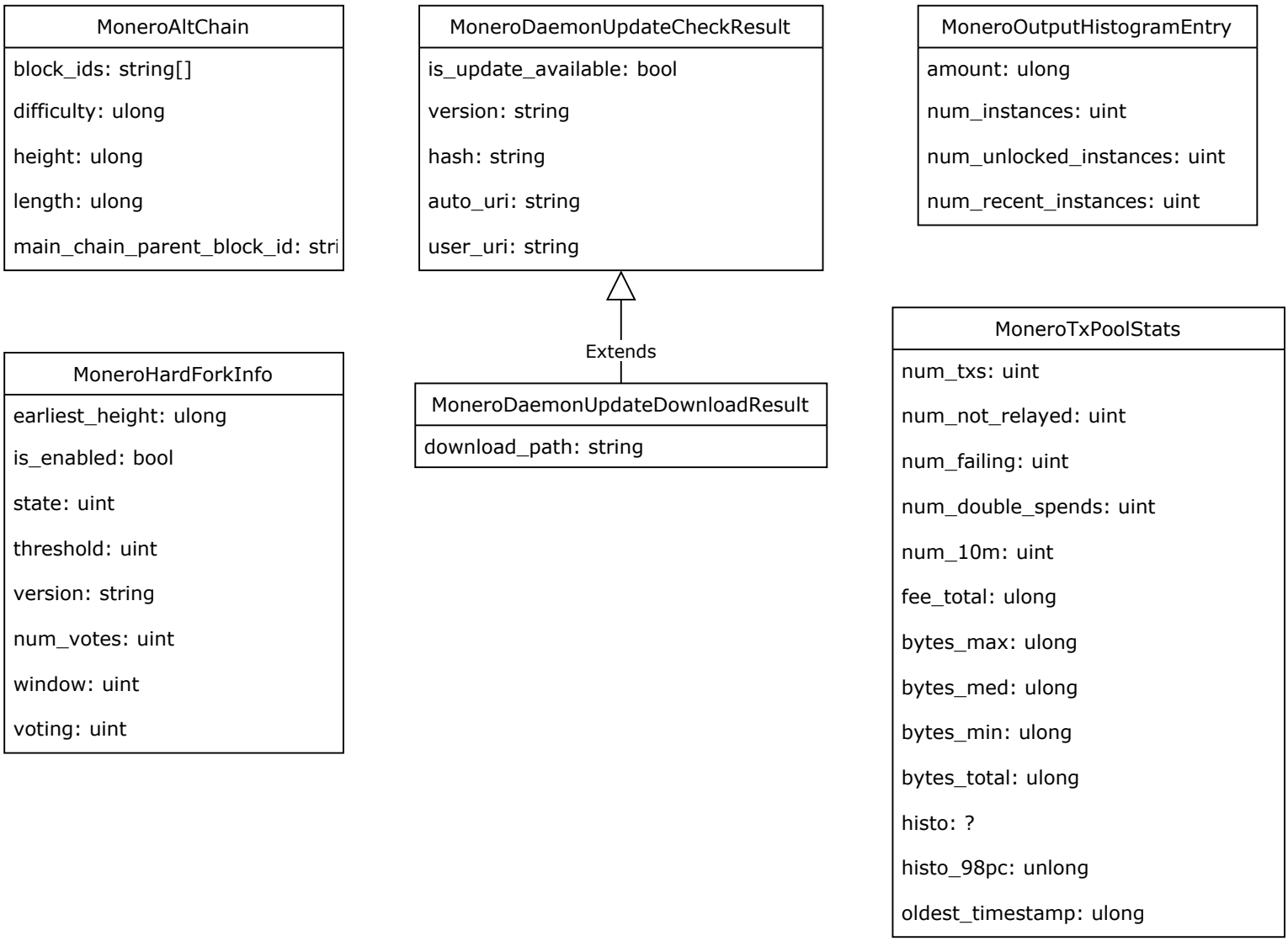
MoneroBan
host: string
ip: string
is_banned: bool
seconds: ulong

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

MoneroMiningStatus
is_active: bool
is_background: bool
address: string
speed: ulong
num_threads: uint

max_used_block_id: string
signatures: string[]

Monero Daemon Types 3/3



Monero Wallet Interface 1/3

getDaemonConnection(): MoneroRpcConnection

getNetworkType(): MoneroNetworkType

getPath(): string

getSeed(): string

getMnemonic(): string

getLanguage(): string

getLanguages(): string[]

getPublicViewKey(): string

getPrivateViewKey(): string

getPublicSpendKey(): string

getPrivateSpendKey(): string

getPrimaryAddress(): string

getAddress(uint accountIdx, uint subaddressIdx): string

getAddressIndex(string address): MoneroSubaddress

getHeight(): ulong

getDaemonHeight(): ulong

getDaemonMaxPeerHeight(): ulong

getApproximateChainHeight(): ulong

sync(ulong startHeight=null, MoneroSyncListener listener=null): MoneroSyncResult

startSyncing(): void

stopSyncing(): void

rescanSpent(): void

rescanBlockchain(): void

getBalance(uint accountIdx=null, uint subaddressIdx=null): ulong

getUnlockedBalance(uint accountIdx=null, uint subaddressIdx=null): ulong

getAccounts(bool includeSubaddresses=false, string tag=null): MoneroAccount[]

getAccount(uint accountIdx, bool includeSubaddresses=false): MoneroAccount

createAccount(string label=null): void

getSubaddresses(uint accountIdx, uint[] subaddressIndices): void

getSubaddress(uint accountIdx, uint subaddressIdx): MoneroSubaddress

createSubaddress(uint accountIdx, string label=null): MoneroSubaddress

getTx(string txId): MoneroTxWallet

getTxs(MoneroTxQuery query=null): MoneroTxWallet[]

getTransfers(uint accountIdx=null, uint subaddressIdx=null): MoneroTransfer[]

getTransfers(MoneroTransferQuery query=null): void

getIncomingTransfers(MoneroTransferQuery query=null): MoneroIncomingTransfer[]

getOutgoingTransfers(MoneroTransferQuery query=null): MoneroOutgoingTransfer[]

getOutputs(MoneroOutputQuery query=null): MoneroOutputWallet[]

Monero Wallet Interface 2/3

getKeyImages(): MoneroKeyImage[]

importOutputsHex(string outputsHex): uint

getOutputsHex(): string

importKeyImages(MoneroKeyImage[] keyImages): MoneroKeyImageImportResult

getNewKeyImagesFromLastImport(): MoneroKeyImage[]

createTx(MoneroSendRequest request): MoneroTxSet

createTx(uint accountIdx, string address, ulong amount): MoneroTxSet

createTxs(MoneroSendRequest request): MoneroTxSet

relayTx(string txMetadata): string

relayTx(MoneroTxWallet tx): string

relayTxs(string[] txMetadatas): string[]

relayTxs(MoneroTxWallet[] txs): string[]

send(MoneroSendRequest request): MoneroTxSet

send(uint accountIdx, string address, ulong amount): MoneroTxSet

sendSplit(MoneroSendRequest request): MoneroTxSet

sendSplit(uint accountIdx, string address, ulong amount): MoneroTxSet

sweepOutput(MoneroSendRequest request): MoneroTxSet

sweepOutput(string address, string keyImage): MoneroTxSet

sweepSubaddress(uint accountIdx, uint subaddressIdx, string address): MoneroTxSet

sweepAccount(uint accountIdx, string address): MoneroTxSet

sweepWallet(string address): MoneroTxSet[]

sweepUnlocked(MoneroSendRequest request): MoneroTxSet[]

sweepDust(bool doNotRelay=false): MoneroTxSet

sign(string message): string

verify(string message, string address, string signature): bool

getTxKey(string txId): string

checkTxKey(string txId, string txKey, string address): MoneroCheckTx

getTxProof(String txId, string address, string message=null): string

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

getSpendProof(string txId, string message=null): string

checkSpendProof(string txId, string message, string signature): bool

getReserveProofWallet(string message): string

getReserveProofAccount(uint accountIdx, ulong amount, string message): string

checkReserveProof(string address, string message, string signature): MoneroCheckReserve

getTxNote(string txId): string

getTxNotes(string[] txIds): string[]

setTxNote(string txId, string note): void

setTxNotes(string[] txIds, string[] notes): void

Monero Wallet Interface 3/3

getAddressBookEntries(uint[] entryIndices=null): MoneroAddressBookEntry[]

deleteAddressBookEntry(uint entryIdx): void

addAddressBookEntry(string address, string description, string paymentId=null): uint

tagAccounts(string tag, uint[] accountIndices): void

untagAccounts(uint[] accountIndices): void

getAccountTags(): MoneroAccountTag[]

setAccountTagLabel(string tag, string label): void

createPaymentUri(MoneroSendRequest request): string

parsePaymentUri(string uri): MoneroSendRequest

getAttribute(string key): string

setAttribute(string key, string val): void

startMining(uint numThreads=null, bool backgroundMining=false, bool ignoreBattery=true): void

stopMining(): void

isMultisigImportNeeded(): bool

isMultisig(): bool

getMultisigInfo(): MoneroMultisigInfo

prepareMultisig(): string

makeMultisig(string[] multisigHexes, uint threshold, string password): MoneroMultisigInitResult

exchangeMultisigKeys(string[] multisigHexes, string password): MoneroMultisigInitResult

getMultisigHex(): string

importMultisigHex(string[] multisigHexes): uint

signMultisigTxHex(string multsiigTxHex): MoneroMultisigSignResult

submitMultisig(string signedMultisigHex): string[]

moveTo(string path, string password): void

save(): void

close(bool save=false): void

getIntegratedAddress(string paymentId=null): MoneroIntegratedAddress (deprecated)

decodeIntegratedAddress(string integratedAddress): MoneroIntegratedAddress (deprecated)

Monero Wallet Types 1/3 - Send Request & Transaction, Transfer, and Output Queries

MoneroSendRequest
destinations: MoneroDestination[]
payment_id: string
priority: MoneroSendPriority
mixin: uint
ring_size: uint
fee: ulong
account_index: uint
subaddress_indices: uint[]
unlock_time: ulong
can_split: bool
do_not_relay: bool
note: string
recipient_name: string
below_amount: ulong
sweep_each_subaddress: bool
key_image: string

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

MoneroTxQuery extends MoneroTxWallet
is_outgoing: bool
is_incoming: bool
tx_ids: string[]
has_payment_id: bool
payment_ids: string[]
height: ulong
min_height: ulong
max_height: ulong
include_outputs: bool
transfer_request: MoneroTransferQuery
output_request: MoneroOutputQuery

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

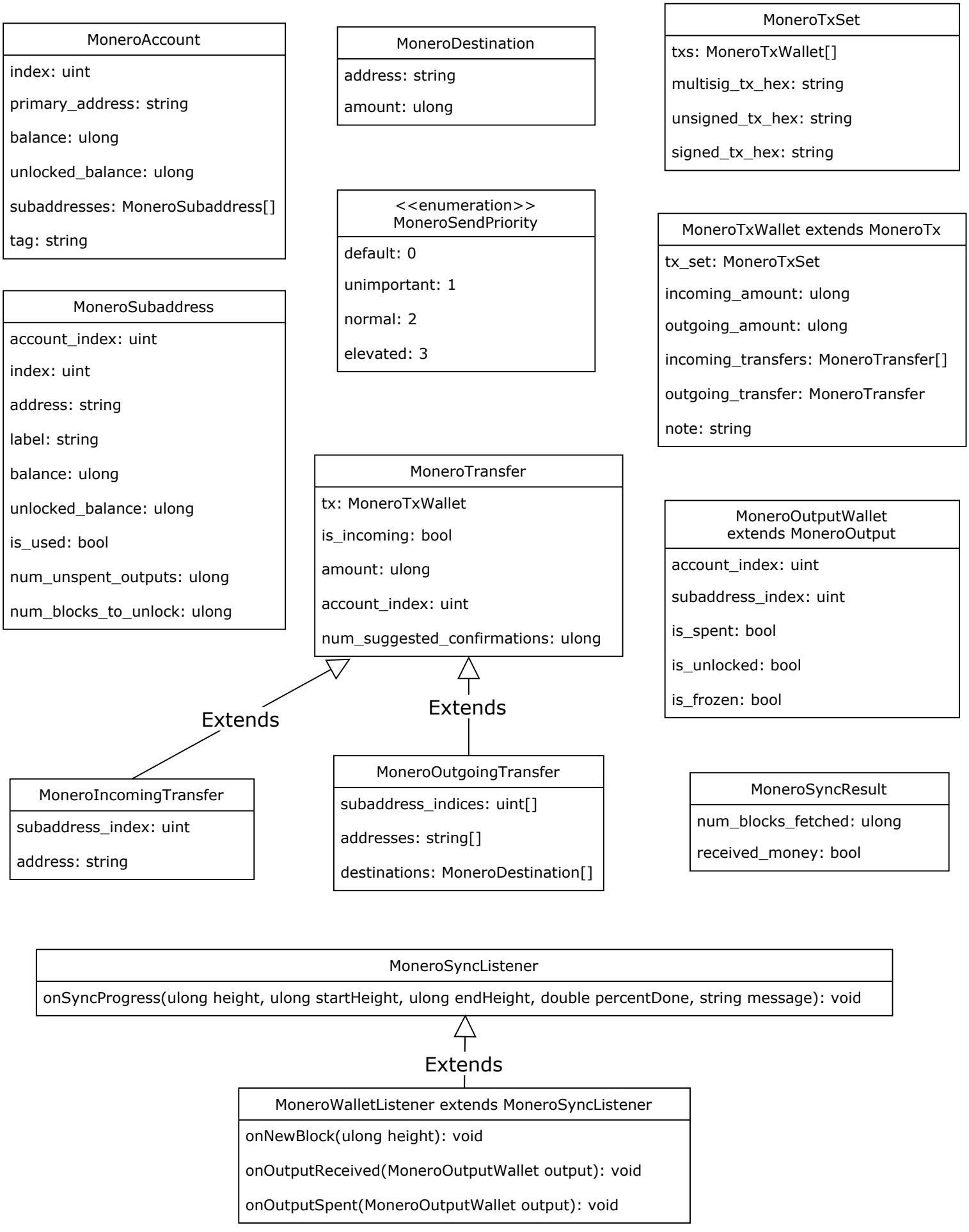
MoneroTransferQuery extends MoneroTransfer
is_incoming: bool
address: string
addresses: string[]
subaddress_index: uint
subaddress_indices: uint[]
destinations: MoneroDestination[]
has_destinations: bool
tx_request: MoneroTxQuery

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

MoneroOutputQuery extends MoneroOutputWallet
subaddress_indices: uint[]
tx_request: MoneroTxRequest

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

Monero Wallet Types 2/3



Monero Wallet Types 3/3

MoneroAccountTag
tag: string
label: string
account_indices: uint[]

MoneroMultisigInfo
is_multisig: bool
is_ready: bool
threshold: uint
num_participants: uint

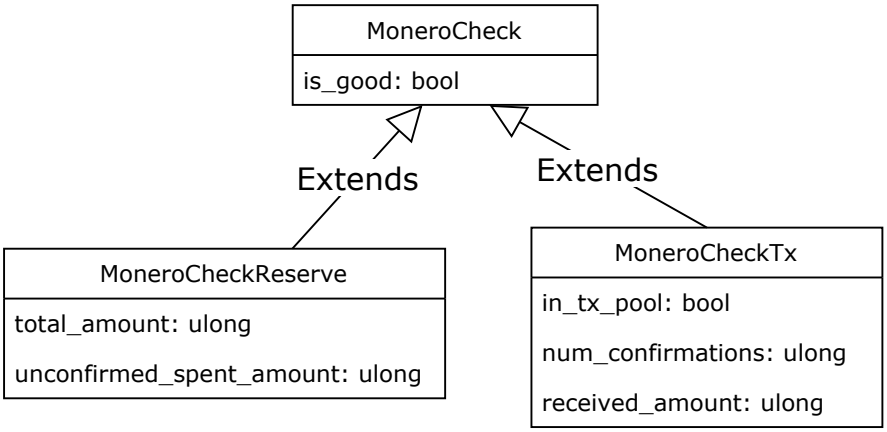
MoneroKeyImageImportResult
height: ulong
spent_amount: ulong
unspent_amount: ulong

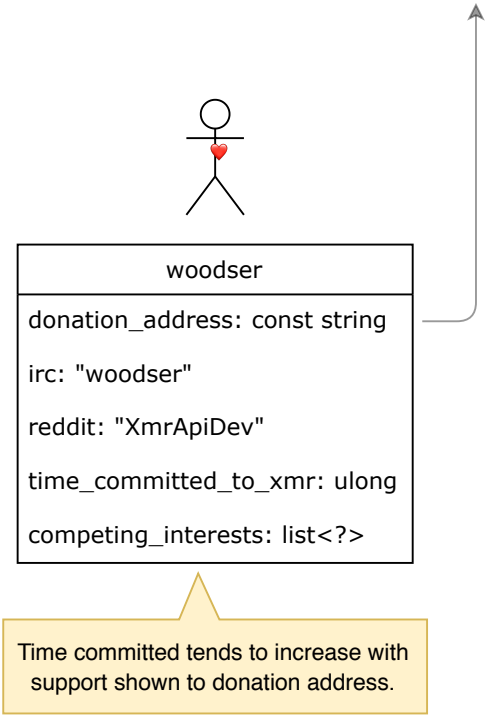
MoneroAddressBookEntry
index: uint
address: string
payment_id: string
description: string

MoneroMultisigInitResult
address: string
multisig_hex: string

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

MoneroMultisigSignResult
signed_multisig_tx_hex: string
tx_ids: string[]



woodser API 1/1

```
46FR1GKVqFNQnDiFkH7AuzbUBrGQwz2VdaXTDD4jcjRE8YkkoTYTmZ2Vohsz9gLSqkj5EM6ai9Q7sBoX4FPPYJdGKQQXPVz
```