GDK Documentation

Release 0.0.30

Blockstream Corporation

Jul 08, 2021

Contents:

1 Gdk Functions **1**

2 Gdk JSON **17**

2.1 Initialization config JSON . . . 17
2.2 Connection parameter JSON . . . 17
2.3 HW device JSON . . . 17
2.4 PIN data JSON . . . 18
2.5 Subaccount detail JSON . . . 18
2.6 Subaccount JSON . . . 18
2.7 Subaccounts list JSON . . . 18
2.8 Transactions list JSON . . . 19
2.9 Transaction details JSON . . . 20
2.10 Create Transaction JSON . . . 21
2.11 Sign Transaction JSON . . . 21
2.12 Send Transaction JSON . . . 23
2.13 Fee Estimates JSON . . . 26
2.14 Two-Factor Config JSON . . . 26
2.15 Settings JSON . . . 27
2.16 Balance Details JSON . . . 28
2.17 Receive Address Details JSON . . . 28
2.18 Receive Address JSON . . . 28
2.19 Utxos details JSON . . . 28
2.20 Transactions Details JSON . . . 28
2.21 Network JSON . . . 28
2.22 Networks list JSON . . . 29
2.23 Transaction Limits JSON . . . 35
2.24 Two-factor detail JSON . . . 35
2.25 Two-factor status JSON . . . 36
2.26 Reconnect hint JSON . . . 36
2.27 Convert data JSON . . . 36
2.28 Balance data JSON . . . 36
2.29 Available currencies JSON . . . 36
2.30 Session event notification JSON . . . 37
2.31 HTTP params JSON . . . 37
2.32 Proxy connectivity params JSON . . . 37
2.33 Locktime Details JSON . . . 37

2.34 Deposit Details JSON 37
2.35 Assets params JSON 37

3 Indices and tables **39**

Index **41**

CHAPTER 1

Gdk Functions

int **GA_init** (const GA_json* *config*)

Set the global configuration and run one-time initialization code. This function must be called once and only once before calling any other functions. When used in a multi-threaded context this function should be called before starting any other threads that call other gdk functions.

Parameters

- **config** – The *Initialization config JSON*.

Returns GA_OK or an error code.

Return type int

int **GA_create_session** (struct GA_session** *session*)

Create a new session.

Parameters

- **session** – Destination for the resulting session. Returned session should be freed using *GA_destroy_session*.

Returns GA_OK or an error code.

Return type int

int **GA_destroy_session** (struct GA_session* *session*)

Free a session allocated by *GA_create_session*.

Parameters

- **session** – Session to free.

Returns GA_OK or an error code.

Return type int

int **GA_connect** (struct GA_session* *session*, const GA_json* *net_params*)

Connect to a remote server using the specified network.

Parameters

- **session** – The session to use.
- **net_params** – The *Connection parameter JSON* of the network to connect to.

Returns GA_OK or an error code.

Return type int

int **GA_disconnect**(struct GA_session* *session*)
Disconnect from a connected remote server.

Parameters

- **session** – The session to use.

Returns GA_OK or an error code.

Return type int

int **GA_reconnect_hint**(struct GA_session* *session*, const GA_json* *hint*)
Configure networking behaviour when reconnecting.

Parameters

- **session** – The session to use.
- **hint** – the *Reconnect hint JSON* to configure.

Returns GA_OK or an error code.

Return type int

int **GA_get_tor_socks5**(struct GA_session* *session*, char** *socks5*)
Get the current SOCKS5 url for the embedded Tor daemon, if any.

Parameters

- **session** – The session to use.
- **socks5** – Destination for the SOCKS5 url (host:port). Empty string if not set. Returned string should be freed using *GA_destroy_string*.

Returns GA_OK or an error code.

Return type int

int **GA_check_proxy_connectivity**(const GA_json* *params*)
Check if server can be reached via the proxy.

Parameters

- **params** – the *Proxy connectivity params JSON* of the server to connect to.

Returns GA_OK or an error code.

Return type int

int **GA_http_get**(struct GA_session* *session*, const GA_json* *params*, GA_json** *output*)
Get JSON data from an https server.

Parameters

- **session** – The session to use.
- **params** – the *HTTP params JSON* of the server to connect to.
- **output** – Destination for the output JSON. Returned GA_json should be freed using *GA_destroy_json*.

Returns GA_OK or an error code.

Return type int

int **GA_refresh_assets**(struct GA_session* *session*, const GA_json* *params*, GA_json** *output*)
Refresh the internal cache asset information.

Parameters

- **session** – The session to use.
- **params** – the *Assets params JSON* of the server to connect to.
- **output** – Destination for the assets JSON. Returned GA_json should be freed using *GA_destroy_json*.

Returns GA_OK or an error code.

Return type int

int **GA_validate_asset_domain_name**(struct GA_session* *session*, const GA_json* *params*, GA_json** *output*)
Validate asset domain name. (This is a interface stub)

Returns GA_OK or an error code.

Return type int

int **GA_register_user**(struct GA_session* *session*, const GA_json* *hw_device*, const char* *mnemonic*, struct GA_auth_handler** *call*)
Create a new user account using a hardware wallet/HSM/TPM.

Parameters

- **session** – The session to use.
- **hw_device** – Details about the *HW device JSON* being used to register.
- **mnemonic** – The user's mnemonic passphrase.
- **call** – Destination for the resulting GA_auth_handler to perform the registration. Returned GA_auth_handler should be freed using *GA_destroy_auth_handler*.

Returns GA_OK or an error code.

Return type int

int **GA_login**(struct GA_session* *session*, const GA_json* *hw_device*, const char* *mnemonic*, const char* *password*, struct GA_auth_handler** *call*)
Authenticate a user using a hardware wallet/HSM/TPM.

Parameters

- **session** – The session to use.
- **hw_device** – Details about the *HW device JSON* being used to login.
- **mnemonic** – The user's mnemonic passphrase.
- **password** – The user's password to decrypt a 27 word mnemonic, or a blank string if none.
- **call** – Destination for the resulting GA_auth_handler to perform the login. Returned GA_auth_handler should be freed using *GA_destroy_auth_handler*.

Returns GA_OK or an error code.

Return type int

int **GA_login_with_pin**(struct GA_session* *session*, const char* *pin*, const GA_json* *pin_data*)
Authenticate a user.

Parameters

- **session** – The session to use.
- **pin** – The user PIN.
- **pin_data** – The *PIN data JSON* returned by *GA_set_pin*.

Returns GA_OK or an error code.

Return type int

int **GA_set_watch_only**(struct GA_session* *session*, const char* *username*, const char* *password*)
Set a watch-only login for the wallet.

Parameters

- **session** – The session to use.
- **username** – The username.
- **password** – The password.

Returns GA_OK or an error code.

Return type int

int **GA_get_watch_only_username**(struct GA_session* *session*, char** *username*)
Get the current watch-only login for the wallet, if any.

Parameters

- **session** – The session to use.
- **username** – Destination for the watch-only username. Empty string if not set. Returned string should be freed using *GA_destroy_string*.

Returns GA_OK or an error code.

Return type int

int **GA_login_watch_only**(struct GA_session* *session*, const char* *username*, const char* *password*)
Authenticate a user in watch only mode.

Parameters

- **session** – The session to use.
- **username** – The username.
- **password** – The password.

Returns GA_OK or an error code.

Return type int

int **GA_remove_account**(struct GA_session* *session*, struct GA_auth_handler** *call*)
Remove an account.

Parameters

- **session** – The session to use.
- **call** – Destination for the resulting GA_auth_handler to perform the removal. Returned GA_auth_handler should be freed using *GA_destroy_auth_handler*.

Returns GA_OK or an error code.

Return type int

int **GA_create_subaccount**(struct GA_session* *session*, const GA_json* *details*, struct GA_auth_handler** *call*)

Create a subaccount.

Parameters

- **session** – The session to use.
- **details** – The *Subaccount detail JSON*. "name" (which must not be already used in the wallet) and "type" (either "2of2" or "2of3") must be populated. For type "2of3" the caller may provide either "recovery_mnemonic" or "recovery_xpub" if they do not wish to have a mnemonic passphrase generated automatically. All other fields are ignored.
- **subaccount** – Destination for the created subaccount details. For 2of3 subaccounts the field "recovery_xpub" will be populated, and "recovery_mnemonic" will contain the recovery mnemonic passphrase if one was generated. These values should be stored safely by the caller as they will not be returned again by any GDK call such as GA_get_subaccounts.
- **call** – Destination for the resulting GA_auth_handler to perform the creation. Returned GA_auth_handler should be freed using *GA_destroy_auth_handler*.

Returns GA_OK or an error code.

Return type int

int **GA_get_subaccounts**(struct GA_session* *session*, struct GA_auth_handler** *call*)

Get the user's subaccount details.

Parameters

- **session** – The session to use.
- **call** – Destination for the resulting GA_auth_handler to perform the creation. Returned GA_auth_handler should be freed using *GA_destroy_auth_handler*.

Returns GA_OK or an error code.

Return type int

int **GA_get_subaccount**(struct GA_session* *session*, uint32_t *subaccount*, struct GA_auth_handler** *call*)

Get subaccount details.

Parameters

- **session** – The session to use.
- **subaccount** – The value of "pointer" from *Subaccounts list JSON* for the subaccount.
- **call** – Destination for the resulting GA_auth_handler to perform the creation. Returned GA_auth_handler should be freed using *GA_destroy_auth_handler*.

Returns GA_OK or an error code.

Return type int

int **GA_rename_subaccount**(struct GA_session* *session*, uint32_t *subaccount*, const char* *new_name*)

Rename a subaccount.

Parameters

- **session** – The session to use.

- **subaccount** – The value of "pointer" from *Subaccounts list JSON* or *Subaccount JSON* for the subaccount to rename.
- **new_name** – New name for the subaccount.

Returns GA_OK or an error code.

Return type int

int **GA_get_transactions**(struct GA_session* *session*, const GA_json* *details*, struct GA_auth_handler** *call*)

Get a page of the user's transaction history.

Parameters

- **session** – The session to use.
- **details** – *Transactions Details JSON* giving the details to get the transactions for.
- **call** – Destination for the resulting GA_auth_handler to complete the action. Returned GA_auth_handler should be freed using `GA_destroy_auth_handler`.

Note: Transactions are returned from newest to oldest with up to 30 transactions per page.

Returns GA_OK or an error code.

Return type int

int **GA_get_receive_address**(struct GA_session* *session*, const GA_json* *details*, struct GA_auth_handler** *call*)

Get a new address to receive coins to.

Parameters

- **session** – The session to use.
- **details** – *Receive Address Details JSON*.
- **call** – Destination for the resulting GA_auth_handler to complete the action. Returned GA_auth_handler should be freed using `GA_destroy_auth_handler`.

Returns GA_OK or an error code.

Return type int

int **GA_get_unspent_outputs**(struct GA_session* *session*, const GA_json* *details*, struct GA_auth_handler** *call*)

Get the user's unspent transaction outputs.

Parameters

- **session** – The session to use.
- **details** – *Utxos details JSON* to get the unspent transaction outputs for.
- **call** – Destination for the resulting GA_auth_handler to complete the action. Returned GA_auth_handler should be freed using `GA_destroy_auth_handler`.

Returns GA_OK or an error code.

Return type int

int **GA_get_unspent_outputs_for_private_key**(struct GA_session* *session*, const char* *private_key*, const char* *password*, uint32_t *unused*, GA_json** *utxos*)

Get the unspent transaction outputs associated with a non-wallet private key.

Parameters

- **session** – The session to use.
- **key** – The private key in WIF or BIP 38 format.
- **password** – The password the key is encrypted with, if any.
- **unused** – unused, must be 0
- **utxos** – Destination for the returned utxos (same format as *Transactions list JSON*). Returned GA_json should be freed using `GA_destroy_json`.

Note: Neither the private key or its derived public key are transmitted.

Returns GA_OK or an error code.

Return type int

int **GA_get_transaction_details**(struct GA_session* *session*, const char* *txhash_hex*, GA_json** *transaction*)

Get a transaction's details.

Parameters

- **session** – The session to use.
- **txhash_hex** – The transaction hash of the transaction to fetch.
- **transaction** – Destination for the *Transaction details JSON*. Returned GA_json should be freed using `GA_destroy_json`.

Returns GA_OK or an error code.

Return type int

int **GA_get_balance**(struct GA_session* *session*, const GA_json* *details*, struct GA_auth_handler** *call*)

The sum of unspent outputs destined to user's wallet.

Parameters

- **session** – The session to use.
- **details** – *Balance Details JSON* giving the subaccount details to get the balance for.
- **call** – Destination for the resulting GA_auth_handler to complete the action. Returned GA_auth_handler should be freed using `GA_destroy_auth_handler`.

Returns GA_OK or an error code.

Return type int

int **GA_get_available_currencies**(struct GA_session* *session*, GA_json** *currencies*)

The list of allowed currencies for all available pricing sources.

Parameters

- **session** – The session to use.

- **currencies** – The returned list of *Available currencies JSON*. Returned GA_json should be freed using `GA_destroy_json`.

Returns GA_OK or an error code.

Return type int

int **GA_convert_amount**(struct GA_session* *session*, const GA_json* *value_details*, GA_json** *output*)
Convert Fiat to BTC and vice-versa.

Parameters

- **session** – The session to use.
- **value_details** – *Convert data JSON* giving the value to convert.
- **output** – Destination for the converted values *Balance data JSON*. Returned GA_json should be freed using `GA_destroy_json`.

Returns GA_OK or an error code.

Return type int

int **GA_set_pin**(struct GA_session* *session*, const char* *mnemonic*, const char* *pin*, const char* *device_id*, GA_json** *pin_data*)
Set a PIN for the user wallet.

Parameters

- **session** – The session to use.
- **mnemonic** – The user's mnemonic passphrase.
- **pin** – The user PIN.
- **device_id** – The user device identifier.
- **pin_data** – The returned *PIN data JSON* containing the user's encrypted mnemonic passphrase. Returned GA_json should be freed using `GA_destroy_json`.

Returns GA_OK or an error code.

Return type int

int **GA_disable_all_pin_logins**(struct GA_session* *session*)
Disable all PIN logins previously set. After calling this method, user will not be able to login with PIN from any device he previously paired. :return: GA_OK or an error code. :rtype: int

int **GA_create_transaction**(struct GA_session* *session*, const GA_json* *transaction_details*, struct GA_auth_handler** *call*)
Construct a transaction.

Parameters

- **session** – The session to use.
- **transaction_details** – The *Create Transaction JSON* for constructing.
- **call** – Destination for the resulting GA_auth_handler to complete the action. Returned GA_auth_handler should be freed using `GA_destroy_auth_handler`.

Returns GA_OK or an error code.

Return type int

int **GA_sign_transaction**(struct GA_session* *session*, const GA_json* *transaction_details*, struct GA_auth_handler** *call*)
Sign the user's inputs to a transaction.

Parameters

- **session** – The session to use.
- **transaction_details** – The *Sign Transaction JSON* for signing, previously returned from GA_create_transaction.
- **call** – Destination for the resulting GA_auth_handler to perform the signing. Returned GA_auth_handler should be freed using *GA_destroy_auth_handler*.

Returns GA_OK or an error code.

Return type int

int **GA_broadcast_transaction**(struct GA_session* *session*, const char* *transaction_hex*, char** *tx_hash*)

Broadcast a non-Green signed transaction to the P2P network.

Parameters

- **session** – The session to use.
- **transaction_hex** – The signed transaction in hex to broadcast.
- **tx_hash** – Destination for the resulting transactions hash. Returned string should be freed using *GA_destroy_string*.

Returns GA_OK or an error code.

Return type int

int **GA_send_transaction**(struct GA_session* *session*, const GA_json* *transaction_details*, struct GA_auth_handler** *call*)

Send a transaction created by GA_create_transaction and signed by GA_sign_transaction.

Parameters

- **session** – The session to use.
- **transaction_details** – The *Send Transaction JSON* for sending.
- **call** – Destination for the resulting GA_auth_handler to perform the send. Returned GA_auth_handler should be freed using *GA_destroy_auth_handler*.

Returns GA_OK or an error code.

Return type int

int **GA_send_nlocktimes**(struct GA_session* *session*)

Request an email containing the user's nLockTime transactions.

Parameters

- **session** – The session to use.

Returns GA_OK or an error code.

Return type int

int **GA_get_expired_deposits**(struct GA_session* *session*, const GA_json* *deposit_details*, struct GA_auth_handler** *call*)

Return UTXOs that can be spent/or are spendable without two factor authentication after a block value.

Parameters

- **session** – The session to use.
- **deposit_details** – The *Deposit Details JSON* for sending.

- **call** – Destination for the resulting GA_auth_handler to change the locktime. Returned GA_auth_handler should be freed using *GA_destroy_auth_handler*.

Returns GA_OK or an error code.

Return type int

int **GA_set_csvtime**(struct GA_session* *session*, const GA_json* *locktime_details*, struct GA_auth_handler** *call*)

Set the number of blocks after which CSV transactions become spendable without two factor authentication.

Parameters

- **session** – The session to use.
- **locktime_details** – The *Locktime Details JSON* for setting the block value.
- **call** – Destination for the resulting GA_auth_handler to change the locktime. Returned GA_auth_handler should be freed using *GA_destroy_auth_handler*.

Returns GA_OK or an error code.

Return type int

int **GA_set_nlocktime**(struct GA_session* *session*, const GA_json* *locktime_details*, struct GA_auth_handler** *call*)

Set the number of blocks after which nLockTime transactions become spendable without two factor authentication. When this function succeeds, if the user has an email address associated with the wallet, an updated nlocktimes.zip file will be sent via email.

Parameters

- **session** – The session to use.
- **locktime_details** – The *Locktime Details JSON* for setting the block value.
- **call** – Destination for the resulting GA_auth_handler to change the locktime. Returned GA_auth_handler should be freed using *GA_destroy_auth_handler*.

Returns GA_OK or an error code.

Return type int

int **GA_set_transaction_memo**(struct GA_session* *session*, const char* *txhash_hex*, const char* *memo*, uint32_t *memo_type*)

Add a transaction memo to a user's GreenAddress transaction.

Parameters

- **session** – The session to use.
- **txhash_hex** – The transaction hash to associate the memo with.
- **memo** – The memo to set.
- **memo_type** – The type of memo to set, either GA_MEMO_USER or GA_MEMO_BIP70.

Returns GA_OK or an error code.

Return type int

int **GA_get_fee_estimates**(struct GA_session* *session*, GA_json** *estimates*)

Get the current network's fee estimates.

Parameters

- **session** – The session to use.

- **estimates** – Destination for the returned *Fee Estimates JSON*. Returned GA_json should be freed using `GA_destroy_json`.

The estimates are returned as an array of 25 elements. Each element is an integer representing the fee estimate expressed as satoshi per 1000 bytes. The first element is the minimum relay fee as returned by the network, while the remaining elements are the current estimates to use for a transaction to confirm from 1 to 24 blocks.

Returns GA_OK or an error code.

Return type int

int **GA_get_mnemonic_passphrase**(struct GA_session* *session*, const char* *password*, char** *mnemonic*)

Get the user's mnemonic passphrase.

Parameters

- **session** – The session to use.
- **password** – Optional password to encrypt the user's mnemonic passphrase with.
- **mnemonic** – Destination for the user's 24 word mnemonic passphrase. if a non-empty password is given, the returned mnemonic passphrase will be 27 words long and will require the password to use for logging in. Returned string should be freed using `GA_destroy_string`.

Returns GA_OK or an error code.

Return type int

int **GA_get_system_message**(struct GA_session* *session*, char** *message_text*)

Get the latest un-acknowledged system message.

Parameters

- **session** – The session to use.
- **message_text** – The returned UTF-8 encoded message text. Returned string should be freed using `GA_destroy_string`.

Note: If all current messages are acknowledged, an empty string is returned.

Returns GA_OK or an error code.

Return type int

int **GA_ack_system_message**(struct GA_session* *session*, const char* *message_text*, struct GA_auth_handler** *call*)

Sign and acknowledge a system message.

The message text will be signed with a key derived from the wallet master key and the signature sent to the server.

Parameters

- **session** – The session to use.
- **message_text** – UTF-8 encoded message text being acknowledged.
- **call** – Destination for the resulting GA_auth_handler to acknowledge the message. Returned GA_auth_handler should be freed using `GA_destroy_auth_handler`.

Returns GA_OK or an error code.

Return type int

int **GA_get_twofactor_config**(struct GA_session* *session*, GA_json** *config*)
Get the two factor configuration for the current user.

Parameters

- **session** – The session to use.
- **config** – Destination for the returned *Two-Factor Config JSON*. Returned GA_json should be freed using *GA_destroy_json*.

Returns GA_OK or an error code.

Return type int

int **GA_change_settings**(struct GA_session* *session*, const GA_json* *settings*, struct GA_auth_handler** *call*)
Change settings

Parameters

- **session** – The session to use.
- **settings** – The new *Settings JSON* values.
- **call** – Destination for the resulting GA_auth_handler. Returned GA_auth_handler should be freed using *GA_destroy_auth_handler*.

Returns GA_OK or an error code.

Return type int

int **GA_get_settings**(struct GA_session* *session*, GA_json** *settings*)
Get settings

Parameters

- **session** – The session to use.
- **settings** – Destination for the current *Settings JSON*. Returned GA_json should be freed using *GA_destroy_json*.

Returns GA_OK or an error code.

Return type int

int **GA_set_notification_handler**(struct GA_session* *session*, GA_notification_handler *handler*, void* *context*)
Set a handler to be called when notifications arrive.

Parameters

- **session** – The server session to receive notifications for.
- **handler** – The handler to receive notifications.
- **context** – A context pointer to be passed to the handler.

This must be called before GA_connect/GA_connect_with_proxy. Notifications may arrive on different threads so the caller must ensure that shared data is correctly locked within the handler. The GA_json object passed to the caller must be destroyed by the caller using GA_destroy_json. Failing to do so will result in memory leaks. When the session is disconnected/destroyed, a final call will be made to the handler with a *Session event notification JSON* notification.

Returns GA_OK or an error code.

Return type int

int **GA_destroy_json**(GA_json* *json*)
Free a GA_json object.

Parameters

- **json** – GA_json object to free.

Returns GA_OK or an error code.

Return type int

int **GA_auth_handler_get_status**(struct GA_auth_handler* *call*, GA_json** *output*)
Get the status/result of an action requiring authorization.

Parameters

- **call** – The auth_handler whose status is to be queried.
- **output** – Destination for the resulting *Two-factor status JSON*. Returned GA_json should be freed using `GA_destroy_json`.

Methods in the api that may require two factor or hardware authentication to complete return a GA_auth_handler object. This object encapsulates the process of determining whether authentication is required and handling conditions such as re-prompting and re-trying after an incorrect two factor code is entered.

The object acts as a state machine which is stepped through by the caller until the desired action is completed. At each step, the current state can be determined and used to perform the next action required.

Some actions require a sequence of codes and decisions; these are hidden behind the state machine interface so that callers do not need to handle special cases or program their own logic to handle any lower level API differences.

The state machine has the following states, which are returned in the "status" element from GA_auth_handler_get_status():

- "done": The action has been completed successfully. Any data returned from the action is present in the "result" element of the status JSON.
- "error": A non-recoverable error occurred performing the action. The associated error message is given in the status element "error". The auth_handler object should be destroyed and the action restarted from scratch if this state is returned.
- "request_code": Two factor authorization is required. The caller should prompt the user to choose a two factor method from the "methods" element and call GA_auth_handler_request_code() with the selected method.
- "resolve_code": The caller should prompt the user to enter the code from the twofactor method chosen in the "request_code" step, and pass this code to GA_auth_handler_resolve_code().
- "call": Twofactor or hardwre authorization is complete and the caller should call GA_auth_handler_call() to perform the action.

Returns GA_OK or an error code.

Return type int

int **GA_auth_handler_request_code**(struct GA_auth_handler* *call*, const char* *method*)
Request a two factor authentication code to authorize an action.

Parameters

- **call** – The auth_handler representing the action to perform.
- **method** – The selected two factor method to use

Returns GA_OK or an error code.

Return type int

int **GA_auth_handler_resolve_code**(struct GA_auth_handler* *call*, const char* *code*)
Authorize an action by providing its previously requested two factor authentication code.

Parameters

- **call** – The auth_handler representing the action to perform.
- **code** – The two factor authentication code received by the user.

Returns GA_OK or an error code.

Return type int

int **GA_auth_handler_call**(struct GA_auth_handler* *call*)
Perform an action following the completion of authorization.

Parameters

- **call** – The auth_handler representing the action to perform.

Returns GA_OK or an error code.

Return type int

int **GA_destroy_auth_handler**(struct GA_auth_handler* *call*)
Free an auth_handler after use.

Parameters

- **call** – The auth_handler to free.

Returns GA_OK or an error code.

Return type int

int **GA_change_settings_twofactor**(struct GA_session* *session*, const char* *method*, const GA_json* *twofactor_details*, struct GA_auth_handler** *call*)
Enable or disable a two factor authentication method.

Parameters

- **session** – The session to use
- **method** – The two factor method to enable/disable, i.e. “email”, “sms”, “phone”, “gauth”
- **twofactor_details** – The two factor method and associated data such as an email address. *Two-factor detail JSON*
- **call** – Destination for the resulting GA_auth_handler to perform the action Returned GA_auth_handler should be freed using `GA_destroy_auth_handler`.

Returns GA_OK or an error code.

Return type int

int **GA_twofactor_reset**(struct GA_session* *session*, const char* *email*, uint32_t *is_dispute*, struct GA_auth_handler** *call*)
Request to begin the two factor authentication reset process.

Parameters

- **session** – The session to use.
- **email** – The new email address to enable once the reset waiting period expires.

- **is_dispute** – GA_TRUE if the reset request is disputed, GA_FALSE otherwise.
- **call** – Destination for the resulting GA_auth_handler to request the reset. Returned GA_auth_handler should be freed using *GA_destroy_auth_handler*.

Returns GA_OK or an error code.

Return type int

int **GA_twofactor_cancel_reset**(struct GA_session* *session*, struct GA_auth_handler** *call*)

Cancel all outstanding two factor resets and unlock the wallet for normal operation.

Parameters

- **session** – The session to use.
- **call** – Destination for the resulting GA_auth_handler to cancel the reset. Returned GA_auth_handler should be freed using *GA_destroy_auth_handler*.

Returns GA_OK or an error code.

Return type int

int **GA_twofactor_change_limits**(struct GA_session* *session*, const GA_json* *limit_details*, struct GA_auth_handler** *call*)

Change twofactor limits settings.

Parameters

- **session** – The session to use.
- **limit_details** – Details of the new *Transaction Limits JSON*
- **call** – Destination for the resulting GA_auth_handler to perform the change. Returned GA_auth_handler should be freed using *GA_destroy_auth_handler*.

Returns GA_OK or an error code.

Return type int

void **GA_destroy_string**(char* *str*)

Free a string returned by the api.

Parameters

- **str** – The string to free.

Returns GA_OK or an error code.

Return type int

int **GA_get_random_bytes**(size_t *num_bytes*, unsigned char* *output_bytes*, size_t *len*)

Get up to 32 random bytes.

Generate up to 32 random bytes using the same strategy as Bitcoin Core code.

Parameters

- **output_bytes** – bytes output buffer
- **siz** – Number of bytes to return (max. 32)

Returns GA_OK or an error code.

Return type int

int **GA_generate_mnemonic**(char** *output*)

Generate a new random BIP 39 mnemonic.

Parameters

- **output** – The generated mnemonic phrase. Returned string should be freed using *GA_destroy_string*.

Returns GA_OK or an error code.

Return type int

int **GA_validate_mnemonic**(const char* *mnemonic*, uint32_t* *valid*)
Validate a BIP 39 mnemonic.

Parameters

- **mnemonic** – The mnemonic phrase
- **valid** – Destination for the result: GA_TRUE if the mnemonic is valid else GA_FALSE

Returns GA_OK or an error code.

Return type int

int **GA_register_network**(const char* *name*, const GA_json* *network_details*)
Register a network configuration

Parameters

- **name** – The name of the network to register
- **network_details** – The *Network JSON* configuration to register

Any existing configuration with the same name is overwritten. If the provided JSON is empty, any existing configuration for the network is removed.

Returns GA_OK or an error code.

Return type int

int **GA_get_networks**(GA_json** *output*)
Get the available network configurations

Parameters

- **output** – Destination for the *Networks list JSON* Returned GA_json should be freed using *GA_destroy_json*.

Returns GA_OK or an error code.

Return type int

int **GA_get_uniform_uint32_t**(uint32_t *upper_bound*, uint32_t* *output*)
Get a uint32_t in the range 0 to (upper_bound - 1) without bias

Parameters

- **output** – Destination for the generated uint32_t.

Returns GA_OK or an error code.

Return type int

CHAPTER 2

Gdk JSON

In this section there are some example JSON used by the lib

2.1 Initialization config JSON

GDK uses the optional `datadir` to store assets and other data.

```
{
    "datadir": "/path/to/datadir"
}
```

2.2 Connection parameter JSON

```
{
   "name": "testnet",
   "log_level": "info",
   "proxy": "localhost:9150",
   "use_tor": true,
   "user_agent": "green_android v2.33"
}
```

2.3 HW device JSON

```
{
   "device": {
      "name": "Ledger",
      "supports_arbitrary_scripts": true,
```

(continues on next page)

(continued from previous page)

```
        "supports_low_r": false
    }
}
```

2.4 PIN data JSON

```
{
    "encrypted_data":
↪"0b39c1e90ca6adce9ff35d1780de74b91d46261a7cbf2b8d2fdc21528c068c8e2b26e3bf3f6a2a992e0e1ecfad0220343
↪",
    "pin_identifier": "38e2f188-b3a8-4d98-a7f9-6c348cb54cfe",
    "salt": "a99/9Qy6P7ON4Umk2FafVQ=="
}
```

2.5 Subaccount detail JSON

```
{
    "name": "subaccount name",
    "type": "2of2"
}
```

2.6 Subaccount JSON

```
{
  "satoshi": {
    "btc": 2034469
    },
  "has_transactions": true,
  "name": "",
  "pointer": 0,
  "receiving_id": "GA3wd2nqwZ8FVwrB8GBsDDh4v8AtdV",
  "recovery_chain_code": "",
  "recovery_pub_key": "",
  "type": "2of2"
}
```

2.7 Subaccounts list JSON

```
[
  {
    "satoshi": {
      "btc": 2034469
      },
    "has_transactions": true,
    "name": "",
```

(continues on next page)

(continued from previous page)


```
    "pointer": 0,
    "receiving_id": "GA3wd2nqwZ8FVwrB8GBsDDh4v8AtdV",
    "recovery_chain_code": "",
    "recovery_pub_key": "",
    "type": "2of2"
  },
  {
    "satoshi": {
      "btc": 977907
      },
    "has_transactions": true,
    "name": "Nuovo",
    "pointer": 1,
    "receiving_id": "GA36xH9spaXv3HCUcjbh7UEPxf1f6t",
    "recovery_chain_code": "",
    "recovery_pub_key": "",
    "type": "2of2"
  }
]
```


2.8 Transactions list JSON


```
[
  {
    "addressees": [
      ""
    ],
    "block_height": 0,
    "calculated_fee_rate": 1004,
    "can_cpfp": true,
    "can_rbf": false,
    "created_at": "2019-02-27 15:12:04",
    "fee": 206,
    "fee_rate": 1004,
    "has_payment_request": false,
    "inputs": [
      {
        "address": "",
        "address_type": "p2wsh",
        "addressee": "",
        "is_output": false,
        "is_relevant": false,
        "is_spent": true,
        "pointer": 1640,
        "pt_idx": 0,
        "satoshi": 1834469,
        "script_type": 14,
        "subaccount": 0,
        "subtype": 0
      }
    ],
    "instant": false,
    "memo": "",
    "outputs": [
```


(continues on next page)

(continued from previous page)


```
      {
        "address": "2N3GFLkDKXZRNUqBdHN2SDdwFXrc5FKAJ3a",
        "address_type": "p2wsh",
        "addressee": "",
        "is_output": true,
        "is_relevant": true,
        "is_spent": false,
        "pointer": 1,
        "pt_idx": 0,
        "satoshi": 200000,
        "script_type": 14,
        "subaccount": 4,
        "subtype": 0
      },
      {
        "address": "2N8HdRzRsV8fF8jWroeX1Hd6CFTBvUuEZfJ",
        "address_type": "p2wsh",
        "addressee": "",
        "is_output": true,
        "is_relevant": false,
        "is_spent": false,
        "pointer": 1657,
        "pt_idx": 1,
        "satoshi": 1634263,
        "script_type": 14,
        "subaccount": 0,
        "subtype": 0
      }
    ],
    "rbf_optin": true,
    "satoshi": 200000,
    "server_signed": true,
    "transaction":
↪"02000000000101e8052d983019fa66c10f311d04f5d11e8ceb058f2653a0f4f74f82283119a7f10100000023220020b511
↪",
    "transaction_locktime": 1481979,
    "transaction_outputs": [],
    "transaction_size": 370,
    "transaction_version": 2,
    "transaction_vsize": 205,
    "transaction_weight": 820,
    "txhash": "fe50531d94fae597d9e209582a401e62b1f705ace93eca94fe2e42f187456e4a",
    "type": "incoming",
    "user_signed": true,
    "vsize": 205
  }
]
```


2.9 Transaction details JSON


```
{
  "transaction":
↪"02000000000101e8052d983019fa66c10f311d04f5d11e8ceb058f2653a0f4f74f82283119a7f10100000023220020b511
↪",
```


(continues on next page)

(continued from previous page)


```
  "transaction_locktime": 1481979,
  "transaction_outputs": [],
  "transaction_size": 370,
  "transaction_version": 2,
  "transaction_vsize": 205,
  "transaction_weight": 820,
  "txhash": "fe50531d94fae597d9e209582a401e62b1f705ace93eca94fe2e42f187456e4a"
}
```


2.10 Create Transaction JSON


```
{
 "addressees": [
   {
     "address": "bitcoin:2NFHMw7GbqnQ3kTYMrA7MnHiYDyLy4EQH6b?amount=0.001"
   }
 ],
 "subaccount": 0
}

{
 "addressees": [
   {
     "address": "2NFHMw7GbqnQ3kTYMrA7MnHiYDyLy4EQH6b",
     "satoshi": 100000
   }
 ],
 "subaccount": 0,
 "fee_rate": 1000
}
```


2.11 Sign Transaction JSON


```
{
"addressees": [
  {
    "address": "2MtcMpWnde3tf5vfwnHXKBaWuAUS8j89771",
    "bip21-params": null,
    "satoshi": 100000
  }
],
"addressees_read_only": false,
"amount_read_only": false,
"available_total": 4999794,
"calculated_fee_rate": 1000,
"change_address": {
  "address": "2NAvvWUygud1YSdsqTZbnntMRjsbx4RxP3Z",
  "address_type": "p2wsh",
  "branch": 1,
  "pointer": 492,
  "script":
↪"522102da0e5f74219dadbd392dc3157c43c3636e237005e7f3976a338e519901fdf9e32103326c44e51893994677bb43e5
↪",
```


(continues on next page)

(continued from previous page)


```
  "script_type": 14,
  "service_xpub":
↪"tpubEAUTpVqYYmDxumXSPwZEgCRC5HZXagbsATdv3wUMweyDrJY4fVDt89ogtpBxa9ynpXB3AyGen3Ko4S8ewpWkkvQsvYP86
↪",
  "subaccount": 0,
  "subtype": null,
  "user_path": [
    1,
    492
  ]
},
"change_amount": 4889588,
"change_index": 0,
"change_subaccount": 0,
"error": "",
"fee": 206,
"fee_rate": 1000,
"have_change": true,
"is_redeposit": false,
"is_sweep": false,
"network_fee": 0,
"satoshi": 100000,
"send_all": false,
"server_signed": false,
"subaccount": 0,
"transaction":
↪"02000000000101c01365291a12d995d7afc3234f4e86d3e064f175ab5a7d47e631de7f293a9309010000002300000000000
↪",
"transaction_locktime": 1483340,
"transaction_outputs": [
  {
    "address": "2NAvvWUygud1YSdsqTZbnntMRjsbx4RxP3Z",
    "address_type": "p2wsh",
    "branch": 1,
    "is_change": true,
    "pointer": 492,
    "satoshi": 4889588,
    "script": "a914c1fc2f90044f58698bf9c51f3283e25c809ac17d87",
    "script_type": 14,
    "service_xpub":
↪"tpubEAUTpVqYYmDxumXSPwZEgCRC5HZXagbsATdv3wUMweyDrJY4fVDt89ogtpBxa9ynpXB3AyGen3Ko4S8ewpWkkvQsvYP86
↪",
    "subaccount": 0,
    "subtype": null,
    "user_path": [
      1,
      492
    ]
  },
  {
    "address": "2MtcMpWnde3tf5vfwnHXKBaWuAUS8j89771",
    "is_change": false,
    "satoshi": 100000,
    "script": "a9140ef7660003133f69023f0436dc8bcf427941dcf587"
  }
],
"transaction_size": 372,
```


(continues on next page)

(continued from previous page)


```
"transaction_version": 2,
"transaction_vsize": 206,
"transaction_weight": 822,
"used_utxos": [
  0
],
"user_signed": false,
"utxo_strategy": "default",
"utxos": [
  {
    "address_type": "p2wsh",
    "block_height": 1448369,
    "ga_asset_id": 1,
    "pointer": 475,
    "prevout_script":
↪"522103bad7ac76143368781c4ac3e7afbb63cd6b52f2a923c715576804aa1046cabc1a210264f5fa70969907861ebdb2b2
↪",
    "pt_idx": 1,
    "satoshi": 4989794,
    "script_type": 14,
    "sequence": 4294967293,
    "service_xpub":
↪"tpubEAUTpVqYYmDxumXSPwZEgCRC5HZXagbsATdv3wUMweyDrJY4fVDt89ogtpBxa9ynpXB3AyGen3Ko4S8ewpWkkvQsvYP86
↪",
    "subaccount": 0,
    "subtype": 0,
    "txhash": "09933a297fde31e6477d5aab75f164e0d3864e4f23c3afd795d9121a296513c0",
    "user_path": [
      1,
      475
    ]
  },
  {
    "address_type": "p2wsh",
    "block_height": 1448369,
    "ga_asset_id": 1,
    "pointer": 474,
    "pt_idx": 0,
    "satoshi": 10000,
    "script_type": 14,
    "subaccount": 0,
    "subtype": 0,
    "txhash": "09933a297fde31e6477d5aab75f164e0d3864e4f23c3afd795d9121a296513c0"
  }
],
"memo": ""
}
```


2.12 Send Transaction JSON


```
{
"addressees": [
  {
    "address": "2NDwUefHRbbHuGsumAWMbRZUzigrtBYkwrq",
```


(continues on next page)

(continued from previous page)


```
    "bip21-params": null,
    "satoshi": 100000
  }
],
"addressees_read_only": false,
"amount_read_only": false,
"available_total": 4999588,
"calculated_fee_rate": 1281,
"change_address": {
  "address": "2Mtpg961bP6WH9cQvY2qS4rnuceoRBrnutn",
  "address_type": "p2wsh",
  "branch": 1,
  "pointer": 497,
  "script":
↪"52210350683b20cc33983f818c9b50606909622dbc4387a17699e5ae09b9d5d1b3111c21028598a36a99fbda64ff1d942a
↪",
  "script_type": 14,
  "service_xpub":
↪"tpubEAUTpVqYYmDxumXSPwZEgCRC5HZXagbsATdv3wUMweyDrJY4fVDt89ogtpBxa9ynpXB3AyGen3Ko4S8ewpWkkvQsvYP86
↪",
  "subaccount": 0,
  "subtype": null,
  "user_path": [
    1,
    497
  ]
},
"change_amount": 109663,
"change_index": 1,
"change_subaccount": 0,
"error": "",
"fee": 337,
"fee_rate": 1000,
"have_change": true,
"is_redeposit": false,
"is_sweep": false,
"memo": "",
"network_fee": 0,
"satoshi": 100000,
"send_all": false,
"server_signed": false,
"subaccount": 0,
"transaction":
↪"020000000001027ff3490a29a2fe73f07e3d3f8740249d61c0025fdc0819586dd9443bc6a00bd30100000023220020ed1
↪",
"transaction_locktime": 1483342,
"transaction_outputs": [
  {
    "address": "2NDwUefHRbbHuGsumAWMbRZUzigrtBYkwrq",
    "is_change": false,
    "satoshi": 100000,
    "script": "a914e2ff64a1ca976947d47b6b2d214af96d5942e1b287"
  },
  {
    "address": "2Mtpg961bP6WH9cQvY2qS4rnuceoRBrnutn",
    "address_type": "p2wsh",
    "branch": 1,
```


(continues on next page)

(continued from previous page)


```
    "is_change": true,
    "pointer": 497,
    "satoshi": 109663,
    "script": "a914114baed477ca8fb65f856b96f860acc52619a6fc87",
    "script_type": 14,
    "service_xpub":
↪"tpubEAUTpVqYYmDxumXSPwZEgCRC5HZXagbsATdv3wUMweyDrJY4fVDt89ogtpBxa9ynpXB3AyGen3Ko4S8ewpWkkvQsvYP86
↪",
    "subaccount": 0,
    "subtype": null,
    "user_path": [
      1,
      497
    ]
  }
],
"transaction_size": 374,
"transaction_version": 2,
"transaction_vsize": 263,
"transaction_weight": 1052,
"used_utxos": [
  1,
  0
],
"user_signed": true,
"utxo_strategy": "default",
"utxos": [
  {
    "address_type": "p2wsh",
    "block_height": 1448369,
    "ga_asset_id": 1,
    "pointer": 474,
    "prevout_script":
↪"522102ff54a17dc6efe168673dbf679fe97e06b5cdcaf7dea8ab83dc6732350cd1b4e4210279979574e0743b4659093c00
↪",
    "pt_idx": 0,
    "satoshi": 10000,
    "script_type": 14,
    "sequence": 4294967293,
    "service_xpub":
↪"tpubEAUTpVqYYmDxumXSPwZEgCRC5HZXagbsATdv3wUMweyDrJY4fVDt89ogtpBxa9ynpXB3AyGen3Ko4S8ewpWkkvQsvYP86
↪",
    "subaccount": 0,
    "subtype": 0,
    "txhash": "09933a297fde31e6477d5aab75f164e0d3864e4f23c3afd795d9121a296513c0",
    "user_path": [
      1,
      474
    ]
  },
  {
    "address_type": "p2wsh",
    "block_height": 0,
    "ga_asset_id": 1,
    "pointer": 493,
    "prevout_script":
↪"522102c9465e8b6e98848428b90f21291a19c62fcb20d2dbff76217068219cada5f7a921022e831b15a4faa339ed9a09a6
↪",
```


(continues on next page)

(continued from previous page)


```
    "pt_idx": 1,
    "satoshi": 200000,
    "script_type": 14,
    "sequence": 4294967293,
    "service_xpub":
↪"tpubEAUTpVqYYmDxumXSPwZEgCRC5HZXagbsATdv3wUMweyDrJY4fVDt89ogtpBxa9ynpXB3AyGen3Ko4S8ewpWkkvQsvYP86
↪",
    "subaccount": 0,
    "subtype": 0,
    "txhash": "d30ba0c63b44d96d581908dc5f02c0619d2440873f3d7ef073fea2290a49f37f",
    "user_path": [
      1,
      493
    ]
  },
  {
    "address_type": "p2wsh",
    "block_height": 0,
    "ga_asset_id": 1,
    "pointer": 494,
    "pt_idx": 0,
    "satoshi": 4789588,
    "script_type": 14,
    "subaccount": 0,
    "subtype": 0,
    "txhash": "d30ba0c63b44d96d581908dc5f02c0619d2440873f3d7ef073fea2290a49f37f"
  }
]
}
```


2.13 Fee Estimates JSON


```
{"fees":[1000,10070,10070,10070,3014,3014,3014,2543,2543,2543,2543,2543,2543,1499,
↪1499,1499,1499,1499,1499,1499,1499,1499,1499,1499,1499]}
```


2.14 Two-Factor Config JSON


```
{
 "all_methods": [
   "email",
   "sms",
   "phone",
   "gauth"
 ],
 "any_enabled": true,
 "email": {
   "confirmed": true,
   "data": "test@test.com",
   "enabled": true
 },
 "enabled_methods": [
```


(continues on next page)

(continued from previous page)


```
    "email"
  ],
  "gauth": {
    "confirmed": false,
    "data": "otpauth://totp/Green%20Bitcoin?secret=IZ3SMET5RDWVUSHB4CPTKUWBJM4HSYHO",
    "enabled": false
  },
  "limits": {
    "bits": "5000.00",
    "btc": "0.00500000",
    "fiat": "0.01",
    "fiat_currency": "EUR",
    "fiat_rate": "1.10000000",
    "is_fiat": false,
    "mbtc": "5.00000",
    "satoshi": 500000,
    "sats": "500000",
    "ubtc": "5000.00"
  },
  "phone": {
    "confirmed": false,
    "data": "",
    "enabled": false
  },
  "sms": {
    "confirmed": false,
    "data": "",
    "enabled": false
  },
  "twofactor_reset": {
    "days_remaining": -1,
    "is_active": false,
    "is_disputed": false
  }
}
```


2.15 Settings JSON


```
{
  "altimeout": 10,
  "notifications": {
    "email_incoming": true,
    "email_outgoing": true
  },
  "pgp": "",
  "pricing": {
    "currency": "EUR",
    "exchange": "KRAKEN"
  },
  "required_num_blocks": 12,
  "sound": true,
  "unit": "BTC"
}
```

2.16 Balance Details JSON

```
{"subaccount":4,"num_confs":0}
```

2.17 Receive Address Details JSON

```
{"subaccount":0,"address_type":"csv"}
```

subaccount The value of "pointer" from *Subaccounts list JSON* or *Subaccount JSON* for the subaccount to generate an address for. Default 0.

address_type One of "csv", "p2sh", "p2wsh". Default value depends on wallet settings.

2.18 Receive Address JSON

```
{
  "address": "2N2x4EgizS2w3DUiWYWW9pEf4sGYRfo6PAX",
  "address_type": "p2wsh",
  "branch": 1,
  "pointer": 13,
  "script":
↪"52210338832debc5e15ce143d5cf9241147ac0019e7516d3d9569e04b0e18f3278718921025dfaa85d64963252604e1b1
↪",
  "script_type": 14,
  "subaccount": 0,
  "subtype": null
}
```

2.19 Utxos details JSON

```
{"subaccount":3,"num_confs":0}
```

2.20 Transactions Details JSON

```
{"subaccount":0,"first":0,"count":30}
```

2.21 Network JSON

```
{
  "address_explorer_url": "http://192.168.56.1:8080/address/",
  "bech32_prefix": "tb",
  "default_peers": [
    "192.168.56.1:19000"
```

(continues on next page)

(continued from previous page)


```
  ],
  "development": true,
  "liquid": false,
  "mainnet": false,
  "name": "Regtest",
  "network": "regtest",
  "p2pkh_version": 111,
  "p2sh_version": 196,
  "service_chain_code":
↪"b60befcc619bb1c212732770fe181f2f1aa824ab89f8aab49f2e13e3a56f0f04",
  "service_pubkey":
↪"036307e560072ed6ce0aa5465534fb5c258a2ccfbc257f369e8e7a181b16d897b3",
  "tx_explorer_url": "http://192.168.56.1:8080/tx/",
  "wamp_cert_pins": [],
  "wamp_onion_url": "",
  "wamp_url": "ws://10.0.2.2:8080/v2/ws"
}
```


2.22 Networks list JSON


```
{
  "all_networks": [
    "mainnet",
    "liquid",
    "testnet",
    "localtest",
    "localtest-liquid",
    "regtest"
  ],
  "liquid": {
    "address_explorer_url": "https://blockstream.info/liquid/address/",
    "asset_registry_onion_url": "http://vi5flmr4z3h3luup.onion",
    "asset_registry_url": "https://assets.blockstream.info",
    "bech32_prefix": "ex",
    "blech32_prefix": "lq",
    "blinded_prefix": 12,
    "csv_buckets": [
      25920,
      51840,
      65535
    ],
    "ct_bits": 52,
    "ct_exponent": 0,
    "default_peers": [],
    "development": false,
    "liquid": true,
    "mainnet": true,
    "name": "Liquid",
    "network": "liquid",
    "p2pkh_version": 57,
    "p2sh_version": 39,
    "policy_asset": "6f0279e9ed041c3d710a9f57d0c02928416460c4b722ae3457a11eec381c526d
↪",
    "service_chain_code":
↪"02721cc509aa0c2f4a90628e9da0391b196abeabc6393ed4789dd6222c43c489",
```


(continues on next page)

(continued from previous page)


```
    "service_pubkey":
↪"02c408c3bb8a3d526103fb93246f54897bdd997904d3e18295b49a26965cb41b7f",
    "tx_explorer_url": "https://blockstream.info/liquid/tx/",
    "wamp_cert_pins": [
      "25847d668eb4f04fdd40b12b6b0740c567da7d024308eb6c2c96fe41d9de218d",
      "a74b0c32b65b95fe2c4f8f098947a68b695033bed0b51dd8b984ecae89571bb6"
    ],
    "wamp_cert_roots": [
      "\n-----BEGIN CERTIFICATE-----
↪\nMIIDSjCCAjKgAwIBAgIQRK+wgNajJ7qJMDmGLvhAazANBgkqhkiG9w0BAQUFADA/
↪\nMSQwIgYDVQQKExtEaWdpdGFsIFNpZ25hdHVyZSBUcnVzdCBDby4xFzAVBgNVBAMT\nDkRTVCBSb290IENBIFgzMB4XDTAwMDk
↪IUmTrE4O\nrz5Iy2Xu/
↪NMhD2XSKtkyj4zl93ewEnu1lcCJo6m67XMuegwGMoOifooUMM0RoOEq\nOLl5CjH9UL2AZd+3UWODyOKIYepLYYHsUmu5ouJLGi
↪5WgTe1QLyNau7Fqckh49ZLOMxt+/yUFw\n7BZy1SbsOFU5Q9D8/
↪RhcQPGX69Wam40dutolucbY38EVAjqr2m7xPi71XAicPNaD\naeQQmxkqtilX4+U9m5/
↪wAl0CAwEAAaNCMEAwDwYDVR0TAQH/BAUwAwEB/
↪zAOBgNV\nHQ8BAf8EBAMCAQYwHQYDVR0OBBYEFMSnsaR7LHH62+FLkHX/
↪xBVghYkQMA0GCSqG\nSIb3DQEBBQUAA4IBAQCjGiybFwBcqR7uKGY3Or+Dxz9LwwmglSBd49lZRNI+DT69\nikugdB/
↪OEIKcdBodfpga3csTS7MgROSR6cz8faXbauX+5v3gTt23ADq1cEmv8uXr\nAvHRAosZy5Q6XkjEGB5YGV8eAlrwDPGxrancWYa
↪md2cXjbDaJWFBM5\nJDGFoqgCWjBH4d1QB7wCCZAA62RjYJsWvIjJEubSfZGL+T0yjWW06XyxV3bqxbYo\nOb8VZRzI9neWagqN
↪----END CERTIFICATE-----",
      "\n-----BEGIN CERTIFICATE-----
↪\nMIIFazCCA1OgAwIBAgIRAIIQz7DSQONZRGPgu2OCiwAwDQYJKoZIhvcNAQELBQAw\nTzELMAkGA1UEBhMCVVMxKTAnBgNVBAo
↪vVqbvYATyjb3miGbESTtrFj/RQSa78f0uoxmyF+\n0TM8ukj13Xnfs7j/
↪EvEhmkvBioZxaUpmZmyPfjxwv60pIgbz5MDmgK7iS4+3mX6U\nA5/
↪TR5d8mUgjU+g4rk8Kb4Mu0UlXjIB0ttov0DiNewNwIRt18jA8+o+u3dpjq+sW\nT8KOEUt+zwvo/
↪7V3LvSye0rgTBIlDHCNAymg4VMk7BPZ7hm/
↪ELNKjD+Jo2FR3qyH\nB5T0Y3HsLuJvW5iB4YlcNHlsdu87kGJ55tukmi8mxdAQ4Q7e2RCOFvu396j3x+UC\nB5iPNgiV5+I3lg
↪lJBdiB3QW0KtZB6awBdpUKD9jf1b0SHzUv\nKBds0pjBqAlkd25HN7rOrFleaJ1/
↪ctaJxQZBKT5ZPt0m9STJEadao0xAH0ahmbWn\nOlFuhjuefXKnEgV4We0+UXgVCwOPjdAvBbI+e0ocS3MFEvzG6uBQE3xDk3Szy
↪rOPNk3sgrDQoo//fb4hVC1CLQJ13hef4Y53CI\nrU7m2Ys6xt0nUW7/
↪vGT1M0NPAgMBAAGjQjBAMA4GA1UdDwEB/wQEAwIBBjAPBgNV\nHRMBAf8EBTADAQH/
↪MB0GA1UdDgQWBBR5tFnme7bl5AFzgAiIyBpY9umbbjANBgkq\nhkiG9w0BAQsFAAOCAgEAVR9YqbyyqFDQDLHYGmkgJykIrGF1X
↪V9lZL\nubhzEFnTIZd+50xx+7LSYK05qAvqFyFWhfFQDlnrzuBZ6brJFe+GnY+EgPbk6ZGQ\n3BebYhtF8GaV0nxvwuo77x/
↪Py9auJ/GpsMiu/X1+mvoiBOv/2X/qkSsisRcOj/
↪KK\nNFtY2PwByVS5uCbMiogziUwthDyC3+6WVwW6LLv3xLfHTjuCvjHIInNzktHCgKQ5\nORAzI4JMPJ+GslWYHb4phowim57ia
↪+sKAIuvtd7u+Nxe5AW0wdeRlN8NwdC\njNPElpzVmbUq4JUagEiuTDkHzsxHpFKVK7q4+63SM1N95R1NbdWhscdCb+ZAJzVc\nc
↪1lvh+wjChP4kqKOJ2qxq\n4RgqsahDYVvTH9w7jXbyLeiNdd8XM2w9U/t7y0Ff/
↪9yi0GE44Za4rF2LN9d11TPA\nmRGunUHBcnWEvgJBQl9nJEiU0Zsnvgc/
↪ubhPgXRR4Xq37Z0j4r7g1SgEEzwxA57d\nemyPxgcYxn/eR44/KJ4EBs+lVDR3veyJm+kXQ99b21/
↪+jh5Xos1AnX5iItreGCc=\n-----END CERTIFICATE-----"
    ],
    "wamp_onion_url": "ws://liquidbtc7u746j4.onion/v2/ws",
    "wamp_url": "wss://liquidwss.greenaddress.it/v2/ws"
  },
  "localtest": {
    "address_explorer_url": "",
    "bech32_prefix": "tb",
    "csv_buckets": [
      144,
      4320,
      51840
    ],
    "default_peers": [],
    "development": true,
    "liquid": false,
    "mainnet": false,
```


(continues on next page)

(continued from previous page)


```
    "name": "Localtest",
    "network": "localtest",
    "p2pkh_version": 111,
    "p2sh_version": 196,
    "service_chain_code":
↪"b60befcc619bb1c212732770fe181f2f1aa824ab89f8aab49f2e13e3a56f0f04",
    "service_pubkey":
↪"036307e560072ed6ce0aa5465534fb5c258a2ccfbc257f369e8e7a181b16d897b3",
    "tx_explorer_url": "",
    "wamp_cert_pins": [],
    "wamp_cert_roots": [
      "\n-----BEGIN CERTIFICATE-----
↪\nMIIDSjCCAjKgAwIBAgIQRK+wgNajJ7qJMDmGLvhAazANBgkqhkiG9w0BAQUFADA/
↪\nMSQwIgYDVQQKExtEaWdpdGFsIFNpZ25hdHVyZSBUcnVzdCBDby4xFzAVBgNVBAMT\nDkRTVCBSb290IENBIFgzMB4XDTAwMDk
↪IUmTrE4O\nrz5Iy2Xu/
↪NMhD2XSKtkyj4zl93ewEnu1lcCJo6m67XMuegwGMoOifooUMM0RoOEq\nOLl5CjH9UL2AZd+3UWODyOKIYepLYYHsUmu5ouJLGi
↪5WgTe1QLyNau7Fqckh49ZLOMxt+/yUFw\n7BZy1SbsOFU5Q9D8/
↪RhcQPGX69Wam40dutolucbY38EVAjqr2m7xPi71XAicPNaD\naeQQmxkqtilX4+U9m5/
↪wAl0CAwEAAaNCMEAwDwYDVR0TAQH/BAUwAwEB/
↪zAOBgNV\nHQ8BAf8EBAMCAQYwHQYDVR0OBBYEFMSnsaR7LHH62+FLkHX/
↪xBVghYkQMA0GCSqG\nSIb3DQEBBQUAA4IBAQCjGiybFwBcqR7uKGY3Or+Dxz9LwwmglSBd49lZRNI+DT69\nikugdB/
↪OEIKcdBodfpga3csTS7MgROSR6cz8faXbauX+5v3gTt23ADq1cEmv8uXr\nAvHRAosZy5Q6XkjEGB5YGV8eAlrwDPGxrancWYa
↪md2cXjbDaJWFBM5\nJDGFoqgCWjBH4d1QB7wCCZAA62RjYJsWvIjJEubSfZGL+T0yjWW06XyxV3bqxbYo\nOb8VZRzI9neWagqN
↪----END CERTIFICATE-----",
      "\n-----BEGIN CERTIFICATE-----
↪\nMIIFazCCA1OgAwIBAgIRAIIQz7DSQONZRGPgu2OCiwAwDQYJKoZIhvcNAQELBQAw\nTzELMAkGA1UEBhMCVVMxKTAnBgNVBAo
↪vVqbvYATyjb3miGbESTtrFj/RQSa78f0uoxmyF+\n0TM8ukj13Xnfs7j/
↪EvEhmkvBioZxaUpmZmyPfjxwv60pIgbz5MDmgK7iS4+3mX6U\nA5/
↪TR5d8mUgjU+g4rk8Kb4Mu0UlXjIB0ttov0DiNewNwIRt18jA8+o+u3dpjq+sW\nT8KOEUt+zwvo/
↪7V3LvSye0rgTBIlDHCNAymg4VMk7BPZ7hm/
↪ELNKjD+Jo2FR3qyH\nB5T0Y3HsLuJvW5iB4YlcNHlsdu87kGJ55tukmi8mxdAQ4Q7e2RCOFvu396j3x+UC\nB5iPNgiV5+I3lg
↪lJBdiB3QW0KtZB6awBdpUKD9jf1b0SHzUv\nKBds0pjBqAlkd25HN7rOrFleaJ1/
↪ctaJxQZBKT5ZPt0m9STJEadao0xAH0ahmbWn\nOlFuhjuefXKnEgV4We0+UXgVCwOPjdAvBbI+e0ocS3MFEvzG6uBQE3xDk3Szy
↪rOPNk3sgrDQoo//fb4hVC1CLQJ13hef4Y53CI\nrU7m2Ys6xt0nUW7/
↪vGT1M0NPAgMBAAGjQjBAMA4GA1UdDwEB/wQEAwIBBjAPBgNV\nHRMBAf8EBTADAQH/
↪MB0GA1UdDgQWBBR5tFnme7bl5AFzgAiIyBpY9umbbjANBgkq\nhkiG9w0BAQsFAAOCAgEAVR9YqbyyqFDQDLHYGmkgJykIrGF1X
↪V9lZL\nubhzEFnTIZd+50xx+7LSYK05qAvqFyFWhfFQDlnrzuBZ6brJFe+GnY+EgPbk6ZGQ\n3BebYhtF8GaV0nxvwuo77x/
↪Py9auJ/GpsMiu/X1+mvoiBOv/2X/qkSsisRcOj/
↪KK\nNFtY2PwByVS5uCbMiogziUwthDyC3+6WVwW6LLv3xLfHTjuCvjHIInNzktHCgKQ5\nORAzI4JMPJ+GslWYHb4phowim57ia
↪+sKAIuvtd7u+Nxe5AW0wdeRlN8NwdC\njNPElpzVmbUq4JUagEiuTDkHzsxHpFKVK7q4+63SM1N95R1NbdWhscdCb+ZAJzVc\no
↪1lvh+wjChP4kqKOJ2qxq\n4RgqsahDYVvTH9w7jXbyLeiNdd8XM2w9U/t7y0Ff/
↪9yi0GE44Za4rF2LN9d11TPA\nmRGunUHBcnWEvgJBQl9nJEiU0Zsnvgc/
↪ubhPgXRR4Xq37Z0j4r7g1SgEEzwxA57d\nemyPxgcYxn/eR44/KJ4EBs+lVDR3veyJm+kXQ99b21/
↪+jh5Xos1AnX5iItreGCc=\n-----END CERTIFICATE-----"
    ],
    "wamp_onion_url": "",
    "wamp_url": "ws://localhost:8080/v2/ws"
  },
  "localtest-liquid": {
    "address_explorer_url": "",
    "asset_registry_onion_url": "http://vi5flmr4z3h3luup.onion",
    "asset_registry_url": "https://assets.blockstream.info",
    "bech32_prefix": "tb",
    "blinded_prefix": 4,
    "csv_buckets": [
      144,
      4320,
```


(continues on next page)

(continued from previous page)


```
      25920,
      51840,
      65535
    ],
    "ct_bits": 52,
    "ct_exponent": 0,
    "default_peers": [],
    "development": true,
    "liquid": true,
    "mainnet": false,
    "name": "Localtest Liquid",
    "network": "localtest-liquid",
    "p2pkh_version": 235,
    "p2sh_version": 75,
    "policy_asset": "5ac9f65c0efcc4775e0baec4ec03abdde22473cd3cf33c0419ca290e0751b225
↪",
    "service_chain_code":
↪"b60befcc619bb1c212732770fe181f2f1aa824ab89f8aab49f2e13e3a56f0f04",
    "service_pubkey":
↪"036307e560072ed6ce0aa5465534fb5c258a2ccfbc257f369e8e7a181b16d897b3",
    "tx_explorer_url": "",
    "wamp_cert_pins": [],
    "wamp_cert_roots": [
      "\n-----BEGIN CERTIFICATE-----
↪\nMIIDSjCCAjKgAwIBAgIQRK+wgNajJ7qJMDmGLvhAazANBgkqhkiG9w0BAQUFADA/
↪\nMSQwIgYDVQQKExtEaWdpdGFsIFNpZ25hdHVyZSBUcnVzdCBDby4xFzAVBgNVBAMT\nDkRTVCBSb290IENBIFgzMB4XDTAwMDk
↪IUmTrE4O\nrz5Iy2Xu/
↪NMhD2XSKtkyj4zl93ewEnu1lcCJo6m67XMuegwGMoOifooUMM0RoOEq\nOLl5CjH9UL2AZd+3UWODyOKIYepLYYHsUmu5ouJLGi
↪5WgTe1QLyNau7Fqckh49ZLOMxt+/yUFw\n7BZy1SbsOFU5Q9D8/
↪RhcQPGX69Wam40dutolucbY38EVAjqr2m7xPi71XAicPNaD\naeQQmxkqtilX4+U9m5/
↪wAl0CAwEAAaNCMEAwDwYDVR0TAQH/BAUwAwEB/
↪zAOBgNV\nHQ8BAf8EBAMCAQYwHQYDVR0OBBYEFMSnsaR7LHH62+FLkHX/
↪xBVghYkQMA0GCSqG\nSIb3DQEBBQUAA4IBAQCjGiybFwBcqR7uKGY3Or+Dxz9Lwwmg1SBd491ZRNI+DT69\nikugdB/
↪OEIKcdBodfpga3csTS7MgROSR6cz8faXbauX+5v3gTt23ADq1cEmv8uXr\nAvHRAosZy5Q6XkjEGB5YGV8eAlrwDPGxrancWYa
↪md2cXjbDaJWFBM5\nJDGFoqgCWjBH4d1QB7wCCZAA62RjYJsWvIjJEubSfZGL+T0yjWW06XyxV3bqxbYo\nOb8VZRzI9neWagqN
↪----END CERTIFICATE-----",
      "\n-----BEGIN CERTIFICATE-----
↪\nMIIFazCCA1OgAwIBAgIRAIIQz7DSQONZRGPgu2OCiwAwDQYJKoZIhvcNAQELBQAw\nTzELMAkGA1UEBhMCVVMxKTAnBgNVBAo
↪vVqbvYATyjb3miGbESTtrFj/RQSa78f0uoxmyF+\n0TM8ukj13Xnfs7j/
↪EvEhmkvBioZxaUpmZmyPfjxwv60pIgbz5MDmgK7iS4+3mX6U\nA5/
↪TR5d8mUgjU+g4rk8Kb4Mu0UlXjIB0ttov0DiNewNwIRt18jA8+o+u3dpjq+sW\nT8KOEUt+zwvo/
↪7V3LvSye0rgTBIlDHCNAymg4VMk7BPZ7hm/
↪ELNKjD+Jo2FR3qyH\nB5T0Y3HsLuJvW5iB4YlcNHlsdu87kGJ55tukmi8mxdAQ4Q7e2RCOFvu396j3x+UC\nB5iPNgiV5+I3lg
↪lJBdiB3QW0KtZB6awBdpUKD9jf1b0SHzUv\nKBds0pjBqAlkd25HN7rOrFleaJ1/
↪ctaJxQZBKT5ZPt0m9STJEadao0xAH0ahmbWn\nOlFuhjuefXKnEgV4We0+UXgVCwOPjdAvBbI+e0ocS3MFEvzG6uBQE3xDk3Szy
↪rOPNk3sgrDQoo//fb4hVC1CLQJ13hef4Y53CI\nrU7m2Ys6xt0nUW7/
↪vGT1M0NPAgMBAAGjQjBAMA4GA1UdDwEB/wQEAwIBBjAPBgNV\nHRMBAf8EBTADAQH/
↪MB0GA1UdDgQWBBR5tFnme7bl5AFzgAiIyBpY9umbbjANBgkq\nhkiG9w0BAQsFAAOCAgEAVR9YqbyyqFDQDLHYGmkgJykIrGF1X
↪V9lZL\nubhzEFnTIZd+50xx+7LSYK05qAvqFyFWhfFQDlnrzuBZ6brJFe+GnY+EgPbk6ZGQ\n3BebYhtF8GaV0nxvwuo77x/
↪Py9auJ/GpsMiu/X1+mvoiBOv/2X/qkSsisRcOj/
↪KK\nNFtY2PwByVS5uCbMiogziUwthDyC3+6WVwW6LLv3xLfHTjuCvjHIInNzktHCgKQ5\nORAzI4JMPJ+GslWYHb4phowim57ia
↪+sKAIuvtd7u+Nxe5AW0wdeRlN8NwdC\njNPElpzVmbUq4JUagEiuTDkHzsxHpFKVK7q4+63SM1N95R1NbdWhscdCb+ZAJzVc\nC
↪1lvh+wjChP4kqKOJ2qxq\n4RgqsahDYVvTH9w7jXbyLeiNdd8XM2w9U/t7y0Ff/
↪9yi0GE44Za4rF2LN9d11TPA\nmRGunUHBcnWEvgJBQl9nJEiU0Zsnvgc/
↪ubhPgXRR4Xq37Z0j4r7g1SgEEzwxA57d\nemyPxgcYxn/eR44/KJ4EBs+lVDR3veyJm+kXQ99b21/
↪+jh5Xos1AnX5iItreGCc=\n-----END CERTIFICATE-----"
    ],
```


(continues on next page)

(continued from previous page)


```
    "wamp_onion_url": "",
    "wamp_url": "ws://localhost:8080/v2/ws"
  },
  "mainnet": {
    "address_explorer_url": "https://blockstream.info/address/",
    "bech32_prefix": "bc",
    "csv_buckets": [
      25920,
      51840,
      65535
    ],
    "default_peers": [],
    "development": false,
    "liquid": false,
    "mainnet": true,
    "name": "Bitcoin",
    "network": "mainnet",
    "p2pkh_version": 0,
    "p2sh_version": 5,
    "service_chain_code":
↪"e9a563d68686999af372a33157209c6860fe79197a4dafd9ec1dbaa49523351d",
    "service_pubkey":
↪"0322c5f5c9c4b9d1c3e22ca995e200d724c2d7d8b6953f7b38fddf9296053c961f",
    "tx_explorer_url": "https://blockstream.info/tx/",
    "wamp_cert_pins": [
      "25847d668eb4f04fdd40b12b6b0740c567da7d024308eb6c2c96fe41d9de218d",
      "a74b0c32b65b95fe2c4f8f098947a68b695033bed0b51dd8b984ecae89571bb6"
    ],
    "wamp_cert_roots": [
      "\n-----BEGIN CERTIFICATE-----
↪\nMIIDSjCCAjKgAwIBAgIQRK+wgNajJ7qJMDmGLvhAazANBgkqhkiG9w0BAQUFADA/
↪\nMSQwIgYDVQQKExtEaWdpdGFsIFNpZ25hdHVyZSBUcnVzdCBDby4xFzAVBgNVBAMT\nDkRTVCBSb290IENBIFgzMB4XDTAwMDk
↪IUmTrE4O\nrz5Iy2Xu/
↪NMhD2XSKtkyj4zl93ewEnu1lcCJo6m67XMuegwGMoOifooUMM0RoOEq\nOLl5CjH9UL2AZd+3UWODyOKIYepLYYHsUmu5ouJLGi
↪5WgTe1QLyNau7Fqckh49ZLOMxt+/yUFw\n7BZy1SbsOFU5Q9D8/
↪RhcQPGX69Wam40dutolucbY38EVAjqr2m7xPi71XAicPNaD\naeQQmxkqtilX4+U9m5/
↪wAl0CAwEAAaNCMEAwDwYDVR0TAQH/BAUwAwEB/
↪zAOBgNV\nHQ8BAf8EBAMCAQYwHQYDVR0OBBYEFMSnsaR7LHH62+FLkHX/
↪xBVghYkQMA0GCSqG\nSIb3DQEBBQUAA4IBAQCjGiybFwBcqR7uKGY3Or+Dxz9LwwmglSBd49lZRNI+DT69\nikugdB/
↪OEIKcdBodfpga3csTS7MgROSR6cz8faXbauX+5v3gTt23ADq1cEmv8uXr\nAvHRAosZy5Q6XkjEGB5YGV8eAlrwDPGxrancWYa
↪md2cXjbDaJWFBM5\nJDGFoqgCWjBH4d1QB7wCCZAA62RjYJsWvIjJEubSfZGL+T0yjWW06XyxV3bqxbYo\nOb8VZRzI9neWagqN
↪----END CERTIFICATE-----",
      "\n-----BEGIN CERTIFICATE-----
↪\nMIIFazCCA1OgAwIBAgIRAIIQz7DSQONZRGPgu2OCiwAwDQYJKoZIhvcNAQELBQAw\nTzELMAkGA1UEBhMCVVMxKTAnBgNVBAo
↪vVqbvYATyjb3miGbESTtrFj/RQSa78f0uoxmyF+\n0TM8ukj13Xnfs7j/
↪EvEhmkvBioZxaUpmZmyPfjxwv60pIgbz5MDmgK7iS4+3mX6U\nA5/
↪TR5d8mUgjU+g4rk8Kb4Mu0UlXjIB0ttov0DiNewNwIRt18jA8+o+u3dpjq+sW\nT8KOEUt+zwvo/
↪7V3LvSye0rgTBIlDHCNAymg4VMk7BPZ7hm/
↪ELNKjD+Jo2FR3qyH\nB5T0Y3HsLuJvW5iB4YlcNHlsdu87kGJ55tukmi8mxdAQ4Q7e2RCOFvu396j3x+UC\nB5iPNgiV5+I3lg
↪lJBdiB3QW0KtZB6awBdpUKD9jf1b0SHzUv\nKBds0pjBqAlkd25HN7rOrFleaJ1/
↪ctaJxQZBKT5ZPt0m9STJEadao0xAH0ahmbWn\nOlFuhjuefXKnEgV4We0+UXgVCwOPjdAvBbI+e0ocS3MFEvzG6uBQE3xDk3Sz
↪rOPNk3sgrDQoo//fb4hVC1CLQJ13hef4Y53CI\nrU7m2Ys6xt0nUW7/
↪vGT1M0NPAgMBAAGjQjBAMA4GA1UdDwEB/wQEAwIBBjAPBgNV\nHRMBAf8EBTADAQH/
↪MB0GA1UdDgQWBBR5tFnme7bl5AFzgAiIyBpY9umbbjANBgkq\nhkiG9w0BAQsFAAOCAgEAVR9YqbyyqFDQDLHYGmkgJykIrGF1X
↪V9lZL\nubhzEFnTIZd+50xx+7LSYK05qAvqFyFWhfFQDlnrzuBZ6brJFe+GnY+EgPbk6ZGQ\n3BebYhtF8GaV0nxvwuo77x/
↪Py9auJ/GpsMiu/X1+mvoiBOv/2X/qkSsisRcOj/
↪KK\nNFtY2PwByVS5uCbMiogziUwthDyC3+6WVwW6LLv3xLfHTjuCvjHIInNzktHCgKQ5\nORAzI4JMPJ+GslWYHb4phowim57ia
↪+sKAIuvtd7u+Nxe5AW0wdeRlN8NwdC\njNPElpzVmbUq4JUagEiuTDkHzsxHpFKVK7q4+63SM1N95R1NbdWaE6cdCb+ZAJzVc\nd
↪1lvh+wjChP4kqKOJ2qxq\n4RgqsahDYVvTH9w7jXbyLeiNdd8XM2w9U/t7y0Ff/
↪9yi0GE44Za4rF2LN9d11TPA\nmRGunUHBcnWEvgJBQl9nJEiU0Zsnvgc/
↪ubhPgXRR4Xq37Z0j4r7g1SgEEzwxA57d\nemyPxgcYxn/eR44/KJ4EBs+lVDR3veyJm+kXQ99b21/
↪+jh5Xos1AnX5iItreGCc=\n-----END CERTIFICATE-----"
```


(continues on next page)

(continued from previous page)


```
    ],
    "wamp_onion_url": "ws://s7a4rvc6425y72d2.onion/v2/ws",
    "wamp_url": "wss://prodwss.greenaddress.it/v2/ws"
  },
  "regtest": {
    "address_explorer_url": "http://192.168.56.1:8080/address/",
    "bech32_prefix": "tb",
    "csv_buckets": [
      144,
      4320,
      51840
    ],
    "default_peers": [
      "192.168.56.1:19000"
    ],
    "development": true,
    "liquid": false,
    "mainnet": false,
    "name": "Regtest",
    "network": "regtest",
    "p2pkh_version": 111,
    "p2sh_version": 196,
    "service_chain_code":
↪"b60befcc619bb1c212732770fe181f2f1aa824ab89f8aab49f2e13e3a56f0f04",
    "service_pubkey":
↪"036307e560072ed6ce0aa5465534fb5c258a2ccfbc257f369e8e7a181b16d897b3",
    "tx_explorer_url": "http://192.168.56.1:8080/tx/",
    "wamp_cert_pins": [],
    "wamp_cert_roots": [],
    "wamp_onion_url": "",
    "wamp_url": "ws://10.0.2.2:8080/v2/ws"
  },
  "testnet": {
    "address_explorer_url": "https://blockstream.info/testnet/address/",
    "bech32_prefix": "tb",
    "csv_buckets": [
      144,
      4320,
      51840
    ],
    "default_peers": [],
    "development": false,
    "liquid": false,
    "mainnet": false,
    "name": "Testnet",
    "network": "testnet",
    "p2pkh_version": 111,
    "p2sh_version": 196,
    "service_chain_code":
↪"b60befcc619bb1c212732770fe181f2f1aa824ab89f8aab49f2e13e3a56f0f04",
    "service_pubkey":
↪"036307e560072ed6ce0aa5465534fb5c258a2ccfbc257f369e8e7a181b16d897b3",
    "tx_explorer_url": "https://blockstream.info/testnet/tx/",
    "wamp_cert_pins": [
      "25847d668eb4f04fdd40b12b6b0740c567da7d024308eb6c2c96fe41d9de218d",
      "a74b0c32b65b95fe2c4f8f098947a68b695033bed0b51dd8b984ecae89571bb6"
    ],
```


(continues on next page)

(continued from previous page)


```
    "wamp_cert_roots": [
      "\n-----BEGIN CERTIFICATE-----
↪\nMIIDSjCCAjKgAwIBAgIQRK+wgNajJ7qJMDmGLvhAazANBgkqhkiG9w0BAQUFADA/
↪\nMSQwIgYDVQQKExtEaWdpdGFsIFNpZ25hdHVyZSBUcnVzdCBDby4xFzAVBgNVBAMT\nDkRTVCBSb290IENBIFgzMB4XDTAwMDk
↪IUmTrE4O\nrz5Iy2Xu/
↪NMhD2XSKtkyj4zl93ewEnu1lcCJo6m67XMuegwGMoOifooUMM0RoOEq\nOLl5CjH9UL2AZd+3UWODyOKIYepLYYHsUmu5ouJLGi
↪5WgTe1QLyNau7Fqckh49ZLOMxt+/yUFw\n7BZy1SbsOFU5Q9D8/
↪RhcQPGX69Wam40dutolucbY38EVAjqr2m7xPi71XAicPNaD\naeQQmxkqtilX4+U9m5/
↪wAl0CAwEAAaNCMEAwDwYDVR0TAQH/BAUwAwEB/
↪zAOBgNV\nHQ8BAf8EBAMCAQYwHQYDVR0OBBYEFMSnsaR7LHH62+FLkHX/
↪xBVghYkQMA0GCSqG\nSIb3DQEBBQUAA4IBAQCjGiybFwBcqR7uKGY3Or+Dxz9LwwmglSBd49lZRNI+DT69\nikugdB/
↪OEIKcdBodfpga3csTS7MgROSR6cz8faXbauX+5v3gTt23ADq1cEmv8uXr\nAvHRAosZy5Q6XkjEGB5YGV8eAlrwDPGxrancWYa
↪md2cXjbDaJWFBM5\nJDGFoqgCWjBH4d1QB7wCCZAA62RjYJsWvIjJEubSfZGL+T0yjWW06XyxV3bqxbYo\nOb8VZRzI9neWagqN
↪----END CERTIFICATE-----",
      "\n-----BEGIN CERTIFICATE-----
↪\nMIIFazCCA1OgAwIBAgIRAIIQz7DSQONZRGPgu2OCiwAwDQYJKoZIhvcNAQELBQAw\nTzELMAkGA1UEBhMCVVMxKTAnBgNVBAo
↪vVqbvYATyjb3miGbESTtrFj/RQSa78f0uoxmyF+\n0TM8ukj13Xnfs7j/
↪EvEhmkvBioZxaUpmZmyPfjxwv60pIgbz5MDmgK7iS4+3mX6U\nA5/
↪TR5d8mUgjU+g4rk8Kb4Mu0UlXjIB0ttov0DiNewNwIRt18jA8+o+u3dpjq+sW\nT8KOEUt+zwvo/
↪7V3LvSye0rgTBIlDHCNAymg4VMk7BPZ7hm/
↪ELNKjD+Jo2FR3qyH\nB5T0Y3HsLuJvW5iB4YlcNHlsdu87kGJ55tukmi8mxdAQ4Q7e2RCOFvu396j3x+UC\nB5iPNgiV5+I3lg
↪lJBdiB3QW0KtZB6awBdpUKD9jf1b0SHzUv\nKBds0pjBqAlkd25HN7rOrFleaJ1/
↪ctaJxQZBKT5ZPt0m9STJEadao0xAH0ahmbWn\nOlFuhjuefXKnEgV4We0+UXgVCwOPjdAvBbI+e0ocS3MFEvzG6uBQE3xDk3Szy
↪rOPNk3sgrDQoo//fb4hVC1CLQJ13hef4Y53CI\nrU7m2Ys6xt0nUW7/
↪vGT1M0NPAgMBAAGjQjBAMA4GA1UdDwEB/wQEAwIBBjAPBgNV\nHRMBAf8EBTADAQH/
↪MB0GA1UdDgQWBBR5tFnme7bl5AFzgAiIyBpY9umbbjANBgkq\nhkiG9w0BAQsFAAOCAgEAVR9YqbyyqFDQDLHYGmkgJykIrGF1X
↪V9lZL\nubhzEFnTIZd+50xx+7LSYK05qAvqFyFWhfFQDlnrzuBZ6brJFe+GnY+EgPbk6ZGQ\n3BebYhtF8GaV0nxvwuo77x/
↪Py9auJ/GpsMiu/X1+mvoiBOv/2X/qkSsisRcOj/
↪KK\nNFtY2PwByVS5uCbMiogziUwthDyC3+6WVwW6LLv3xLfHTjuCvjHIInNzktHCgKQ5\nORAzI4JMPJ+GslWYHb4phowim57ia
↪+sKAIuvtd7u+Nxe5AW0wdeRlN8NwdC\njNPElpzVmbUq4JUagEiuTDkHzsxHpFKVK7q4+63SM1N95R1NbdWhscdCb+ZAJzVc\nc
↪1lvh+wjChP4kqKOJ2qxq\n4RgqsahDYVvTH9w7jXbyLeiNdd8XM2w9U/t7y0Ff/
↪9yi0GE44Za4rF2LN9d11TPA\nmRGunUHBcnWEvgJBQl9nJEiU0Zsnvgc/
↪ubhPgXRR4Xq37Z0j4r7g1SgEEzwxA57d\nemyPxgcYxn/eR44/KJ4EBs+lVDR3veyJm+kXQ99b21/
↪+jh5Xos1AnX5iItreGCc=\n-----END CERTIFICATE-----"
    ],
    "wamp_onion_url": "ws://gu5ke7a2aguwfqhz.onion/v2/ws",
    "wamp_url": "wss://testwss.greenaddress.it/v2/ws"
  }
}
```


2.23 Transaction Limits JSON


```
{"is_fiat":false,"mbtc":"555"}
{"is_fiat":true,"fiat":"555"}
```


2.24 Two-factor detail JSON


```
{"confirmed":true,"data":"mail@example.com","enabled":true}
```

2.25 Two-factor status JSON


```
{"action":"disable_2fa","device":null,"methods":["gauth"],"status":"request_code"}
```


2.26 Reconnect hint JSON


```
{ "hint" : "now" }
```

```
{ "hint" : "disable" }
```

```
{ "tor_sleep_hint" : "wakeup", "hint": "start" }
```

```
{ "tor_sleep_hint" : "sleep" }
```


2.27 Convert data JSON


```
{
  "satoshi": 1120
}
```


2.28 Balance data JSON


```
{
  "bits": "20344.69",
  "btc": "0.02034469",
  "fiat": "0.02",
  "fiat_currency": "EUR",
  "fiat_rate": "1.10000000",
  "mbtc": "20.34469",
  "satoshi": 2034469,
  "sats": "2034469",
  "subaccount": 0,
  "ubtc": "20344.69"
}
```


2.29 Available currencies JSON


```
{
  "all":["AUD","BRL","CAD","CHF","CNY","DKK","EUR","GBP","HKD","IDR","INR","JPY","MXN
↪","MYR","NGN","NOK","NZD","PLN","RUB","SEK","SGD","THB","TRY","USD","ZAR"],
  "per_exchange":{"BITFINEX":["USD"],"BITSTAMP":["USD"],"BTCAVG":[],"BTCCHINA":[],
↪"HUOBI":[],"KIWICOIN":["NZD"],"KRAKEN":["EUR","USD"],"LOCALBTC":["AUD","BRL","CAD",
↪"CHF","CNY","DKK","EUR","GBP","HKD","IDR","INR","JPY","MXN","MYR","NGN","NOK","NZD",
↪"PLN","RUB","SEK","SGD","THB","TRY","USD","ZAR"],"LUNO":["IDR","MYR","NGN","ZAR"],
↪"QUADRIGACX":["CAD","USD"],"TRT":["EUR"]}
```


(continues on next page)

(continued from previous page)

```
}
```

2.30 Session event notification JSON

```
{
   "event": "session"
   "session": {"connected": false}
}
```

2.31 HTTP params JSON

```
{
   "uri":"https://assets.blockstream.info"
   "target":"/index.json"
   "proxy":"localhost:9150"
   "headers":{"If-Modified-Since":"Mon, 02 Sep 2019 22:39:39 GMT"}
}
```

2.32 Proxy connectivity params JSON

```
{
   "name":"testnet"
   "use_tor":true
   "proxy":"localhost:9150"
}
```

2.33 Locktime Details JSON

```
{"value":65535}
```

2.34 Deposit Details JSON

```
{"subaccount":3,"num_confs":0, "expires_at_block":65535}
```

2.35 Assets params JSON

```
{
   "assets":True,
   "icons":True,
```

(continues on next page)

(continued from previous page)

```
    "refresh":True
}
```

CHAPTER 3

Indices and tables

- genindex
- search

Index

G

GA_ack_system_message (*C function*), 11
GA_auth_handler_call (*C function*), 14
GA_auth_handler_get_status (*C function*), 13
GA_auth_handler_request_code (*C function*), 13
GA_auth_handler_resolve_code (*C function*), 14
GA_broadcast_transaction (*C function*), 9
GA_change_settings (*C function*), 12
GA_change_settings_twofactor (*C function*), 14
GA_check_proxy_connectivity (*C function*), 2
GA_connect (*C function*), 1
GA_convert_amount (*C function*), 8
GA_create_session (*C function*), 1
GA_create_subaccount (*C function*), 5
GA_create_transaction (*C function*), 8
GA_destroy_auth_handler (*C function*), 14
GA_destroy_json (*C function*), 12
GA_destroy_session (*C function*), 1
GA_destroy_string (*C function*), 15
GA_disable_all_pin_logins (*C function*), 8
GA_disconnect (*C function*), 2
GA_generate_mnemonic (*C function*), 15
GA_get_available_currencies (*C function*), 7
GA_get_balance (*C function*), 7
GA_get_expired_deposits (*C function*), 9
GA_get_fee_estimates (*C function*), 10
GA_get_mnemonic_passphrase (*C function*), 11
GA_get_networks (*C function*), 16
GA_get_random_bytes (*C function*), 15
GA_get_receive_address (*C function*), 6
GA_get_settings (*C function*), 12
GA_get_subaccount (*C function*), 5
GA_get_subaccounts (*C function*), 5
GA_get_system_message (*C function*), 11
GA_get_tor_socks5 (*C function*), 2
GA_get_transaction_details (*C function*), 7
GA_get_transactions (*C function*), 6
GA_get_twofactor_config (*C function*), 12
GA_get_uniform_uint32_t (*C function*), 16
GA_get_unspent_outputs (*C function*), 6
GA_get_unspent_outputs_for_private_key (*C function*), 6
GA_get_watch_only_username (*C function*), 4
GA_http_get (*C function*), 2
GA_init (*C function*), 1
GA_login (*C function*), 3
GA_login_watch_only (*C function*), 4
GA_login_with_pin (*C function*), 3
GA_reconnect_hint (*C function*), 2
GA_refresh_assets (*C function*), 3
GA_register_network (*C function*), 16
GA_register_user (*C function*), 3
GA_remove_account (*C function*), 4
GA_rename_subaccount (*C function*), 5
GA_send_nlocktimes (*C function*), 9
GA_send_transaction (*C function*), 9
GA_set_csvtime (*C function*), 10
GA_set_nlocktime (*C function*), 10
GA_set_notification_handler (*C function*), 12
GA_set_pin (*C function*), 8
GA_set_transaction_memo (*C function*), 10
GA_set_watch_only (*C function*), 4
GA_sign_transaction (*C function*), 8
GA_twofactor_cancel_reset (*C function*), 15
GA_twofactor_change_limits (*C function*), 15
GA_twofactor_reset (*C function*), 14
GA_validate_asset_domain_name (*C function*), 3
GA_validate_mnemonic (*C function*), 16