
GDK Documentation

Release 0.0.55

Blockstream Corporation

Jul 11, 2022

Contents:

1	GDK Functions	1
2	GDK JSON	21
2.1	Initialization config JSON	21
2.2	Connection parameters JSON	21
2.3	Proxy Settings JSON	22
2.4	Login credentials JSON	22
2.5	HW device JSON	23
2.6	PIN data JSON	23
2.7	Encrypt with PIN JSON	24
2.8	Encrypt with PIN Result JSON	24
2.9	Wallet identifier request JSON	24
2.10	Get credentials JSON	24
2.11	Subaccount JSON	25
2.12	Subaccount update JSON	25
2.13	Subaccounts list JSON	26
2.14	Transaction list JSON	26
2.15	Transaction details JSON	27
2.16	Create transaction JSON	28
2.17	Sign transaction JSON	28
2.18	Send transaction JSON	30
2.19	Sign PSBT JSON	33
2.20	Sign PSBT Result JSON	33
2.21	PSBT Get Details JSON	33
2.22	PSBT Get Details Result JSON	34
2.23	Fee estimates JSON	34
2.24	Two-Factor config JSON	34
2.25	Settings JSON	35
2.26	Receive address details JSON	36
2.27	Previous addresses request JSON	37
2.28	Previous addresses JSON	37
2.29	Unspent outputs request JSON	38
2.30	Unspent outputs JSON	39
2.31	Unspent outputs set status JSON	40
2.32	Transactions details JSON	41
2.33	Network JSON	41

2.34	Network list JSON	41
2.35	Transaction limits JSON	42
2.36	Two-factor detail JSON	42
2.37	Auth handler status JSON	42
2.38	Reconnect JSON	43
2.39	Convert amount JSON	44
2.40	Amount JSON	44
2.41	Available currencies JSON	45
2.42	HTTP parameters JSON	45
2.43	Locktime details JSON	45
2.44	Asset parameters JSON	46
2.45	Get assets parameters JSON	46
2.46	Asset informations JSON	46
2.47	Error details JSON	47
2.48	Get Subaccounts parameters JSON	47
3	GDK Notifications	49
3.1	Network notification	49
3.2	Tor notification	49
3.3	Settings notification	50
3.4	Two factor reset notification	50
3.5	Block notification	50
3.6	Transaction notification	51
3.7	Ticker notification	51
4	GDK Hardware Wallet Interface	53
4.1	Required Data JSON	53
4.2	Hardware Get XPubs Action	53
4.3	Hardware Get Master Blinding Key Action	54
4.4	Hardware Sign Message Action	54
4.5	Hardware Get Blinding Public Keys Action	55
4.6	Hardware Get Blinding Nonces Action	55
5	Indices and tables	57
	Index	59

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

Perform one-time initialization of the library. This call must be made once only before calling any other GDK functions, including any functions called from other threads.

Parameters

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

Returns GA_OK or an error code.

Return type int

int **GA_get_thread_error_details** (GA_json** *output*)

Get any error details associated with the last error on the current thread.

Parameters

- **output** – Destination for the output *Error details JSON* JSON. Returned GA_json should be freed using *GA_destroy_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. The returned session should be freed using *GA_destroy_session*.

Once created, the caller should set a handler for notifications using *GA_set_notification_handler*, before calling *GA_connect* to connect the session to the network for use. :return: GA_OK or an error code.
:rtype: 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 for a session.

Parameters

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

This call must be initially made on a session before *GA_connect*. *GDK 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.

Once a session has been connected, this call can be made only with null values for *handler* and *context*. Once this returns, no further notifications will be delivered for the lifetime of the session.

The caller should not call session functions from within the callback handler as this may block the application.
:return: GA_OK or an error code. :rtype: int

int **GA_destroy_session** (struct GA_session* *session*)
Free a session allocated by *GA_create_session*.

Parameters

- **session** – The session to free.

If the session was connected using *GA_connect* then this call will disconnect it before destroying it. :return: GA_OK or an error code. :rtype: int

int **GA_connect** (struct GA_session* *session*, const GA_json* *net_params*)
Connect the session to the specified network.

Parameters

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

This call connects to the remote network services that the session requires, for example the Green servers or Electrum servers. *GA_connect* must be called only once per session lifetime, after *GA_create_session* and before *GA_destroy_session* respectively. Once connected, the underlying network connection of the session can be controlled using *GA_reconnect_hint*.

Once the session is connected, use *GA_register_user* to create a new wallet for the session, or *GA_login_user* to open an existing wallet. :return: GA_OK or an error code. :rtype: int

int **GA_reconnect_hint** (struct GA_session* *session*, const GA_json* *hint*)
Connect or disconnect a sessions underlying network connection.

Parameters

- **session** – The session to use.
- **hint** – the *Reconnect JSON* describing the desired reconnection behaviour.

Returns GA_OK or an error code.

Return type int

int **GA_get_proxy_settings** (struct GA_session* *session*, GA_json** *output*)
Get the current proxy settings for the given session.

Parameters

- **session** – The session to use.

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

Returns GA_OK or an error code.

Return type int

int **GA_get_wallet_identifier** (const GA_json* *net_params*, const GA_json* *params*, GA_json** *output*)

Compute a hashed wallet identifier from a BIP32 xpub or mnemonic.

The identifier returned is computed from the network combined with the master chain code and public key of the xpub/mnemonic. It can be used as a unique wallet identifier to mitigate privacy risks associated with storing the wallet's xpub.

Parameters

- **net_params** – The *Connection parameters JSON* of the network to compute an identifier for.
- **params** – The *Wallet identifier request JSON* to compute an identifier for.
- **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_http_request** (struct GA_session* *session*, const GA_json* *params*, GA_json** *output*)

Make a request to an http server.

Parameters

- **session** – The session to use.
- **params** – the *HTTP parameters 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 sessions internal cache of Liquid asset information.

Parameters

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

Each release of GDK comes with a list of the latest registered Liquid assets built-in. This call is used to return this data and/or to update it to include any new assets that have been registered since installation or the last update. :return: GA_OK or an error code. :rtype: int

int **GA_get_assets** (struct GA_session* *session*, const GA_json* *params*, GA_json** *output*)

Query the Liquid asset registry.

Parameters

- **session** – The session to use.

- **params** – the *Get assets parameters JSON* specifying the assets to query.
- **output** – Destination for the output *Asset informations JSON*. Returned GA_json should be freed using *GA_destroy_json*.

This call is used to retrieve informations about a set of Liquid assets specified by their asset id. :return: GA_OK or an error code. :rtype: 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 GA_json* details,
                     struct GA_auth_handler** call)
```

Create a new user wallet.

Parameters

- **session** – The session to use.
- **hw_device** – *HW device JSON* or empty JSON for software wallet registration.
- **details** – The *Login credentials JSON* for software wallet registration.
- **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_user (struct GA_session* session, const GA_json* hw_device, const GA_json* details, struct
                  GA_auth_handler** call)
```

Authenticate to a user's wallet.

Parameters

- **session** – The session to use.
- **hw_device** – *HW device JSON* or empty JSON for software wallet login.
- **details** – The *Login credentials JSON* for authenticating the user.
- **call** – Destination for the resulting GA_auth_handler to perform the login. Returned GA_auth_handler should be freed using *GA_destroy_auth_handler*.

If a sessions underlying network connection has disconnected and reconnected, the user will need to login again using this function. In this case, the caller can pass empty JSON for both hw_device and details to login using the previously passed credentials and device. :return: GA_OK or an error code. :rtype: int

```
int GA_set_watch_only (struct GA_session* session, const char* username, const char* password)
```

Set or disable a watch-only login for a logged-in user wallet.

Parameters

- **session** – The session to use.
- **username** – The watch-only username to login with, or a blank string to disable.
- **password** – The watch-only password to login with, or a blank string to disable.

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 a logged-in user 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_remove_account** (struct GA_session* *session*, struct GA_auth_handler** *call*)

Remove and delete the server history of a wallet.

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](#).

For multisig Green sessions, removing a wallet removes all history and data associated with the wallet on the server. This operation cannot be undone, and re-registering the wallet will not bring back the wallet's history. For this reason, only empty wallets can be deleted.

For singlesig sessions, removing a wallet removes the locally persisted cache. The actual removal will happen after [GA_destroy_session](#) is called. :return: GA_OK or an error code. :rtype: 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 "name" (which must not be already used in the wallet) and "type" (either "2of2", "2of2_no_recovery" or "2of3") must be populated. Type "2of2_no_recovery" is available only for Liquid networks and always requires both keys for spending. 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.
- **call** – Destination for the resulting GA_auth_handler to perform the creation. Returned GA_auth_handler should be freed using [GA_destroy_auth_handler](#). Details of the created subaccount are returned in the "result" element of the GA_auth_handler. 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 must be stored safely by the caller as they will not be returned again by any call such as [GA_get_subaccounts](#).

Returns GA_OK or an error code.

Return type int

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

Get the user's subaccount details.

Parameters

- **session** – The session to use.

- **details** – the *Get Subaccounts parameters JSON* controlling the request.
- **call** – Destination for the resulting GA_auth_handler to perform the creation. The call handlers result is *Subaccounts list JSON*. 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. The call handlers result is *Subaccount JSON*. 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.

Note: This call is deprecated and will be removed in a future release. Use *GA_update_subaccount* to rename subaccounts.

Returns GA_OK or an error code.

Return type int

```
int GA_update_subaccount (struct GA_session* session, const GA_json* details, struct  
GA_auth_handler** call)  
Update subaccount information.
```

Parameters

- **session** – The session to use.
- **details** – *Subaccount update JSON* giving the details to update.
- **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_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_previous_addresses (struct GA_session* session, const GA_json* details, struct
                        GA_auth_handler** call)
```

Get a page of addresses previously generated for a subaccount.

Parameters

- **session** – The session to use.
- **details** – *Previous addresses request JSON* detailing the previous addresses to fetch.
- **call** – Destination for the resulting GA_auth_handler to complete the action. The call handlers result is *Previous addresses JSON*. Returned GA_auth_handler should be freed using *GA_destroy_auth_handler*.

Note: Iteration of all addresses is complete when 'last_pointer' is not present in the results.

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** – *Unspent outputs request JSON* detailing the unspent transaction outputs to fetch.
- **call** – Destination for the resulting GA_auth_handler to complete the action. The call handlers result is *Unspent outputs JSON*. 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 *Transaction 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_set_unspent_outputs_status** (struct GA_session* session, const GA_json* details, struct GA_auth_handler** call)

Change the status of a user's unspent transaction outputs.

Parameters

- **session** – The session to use.
- **details** – *Unspent outputs set status JSON* detailing the unspent transaction outputs status to set.
- **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_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*)

Get the sum of unspent outputs paying to a subaccount.

Parameters

- **session** – The session to use.
- **details** – *Unspent outputs request JSON* detailing the unspent transaction outputs to compute the balance from.
- **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*)

Get 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 amount JSON* giving the value to convert.
- **output** – Destination for the converted values *Amount JSON*. Returned GA_json should be freed using *GA_destroy_json*.

Returns GA_OK or an error code.

Return type int

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

Encrypt json with server provided key protected by a PIN.

Parameters

- **session** – The session to use.
- **details** – The *Encrypt with PIN JSON* to encrypt.
- **call** – Destination for the resulting GA_auth_handler to complete the action. The call handlers result is *Encrypt with PIN Result JSON* which the caller should persist. Returned GA_auth_handler should be freed using *GA_destroy_auth_handler*.

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, the user will not be able to login with PIN from any device that was previously enabled using [GA_encrypt_with_pin](#).

Parameters

- **session** – The session to use.

Returns GA_OK or an error code.

Return type 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_psbt_sign** (struct GA_session* *session*, const GA_json* *details*, struct GA_auth_handler** *call*)

Sign one or more of a user's inputs in a PSBT or PSET.

Parameters

- **session** – The session to use.
- **details** – The [Sign PSBT JSON](#) for signing.
- **call** – Destination for the resulting GA_auth_handler to perform the signing. The call handlers result is [Sign PSBT Result JSON](#). Returned GA_auth_handler should be freed using [GA_destroy_auth_handler](#).

Note: EXPERIMENTAL warning: this call may be changed in future releases.

Returns GA_OK or an error code.

Return type int

int **GA_psbt_get_details** (struct GA_session* session, const GA_json* details, struct GA_auth_handler** call)
 Get wallet details of a PSBT or PSET.

Parameters

- **session** – The session to use.
- **details** – The *PSBT Get Details JSON* for getting the wallet details.
- **call** – Destination for the resulting GA_auth_handler to get the wallet details. The call handlers result is *PSBT Get Details Result JSON*. Returned GA_auth_handler should be freed using *GA_destroy_auth_handler*.

Note: EXPERIMENTAL warning: this call may be changed in future releases.

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 fully 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_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 call succeeds, if the user has an email address associated with the wallet, an updated nlock-times.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** – Unused, pass 0.

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_credentials (struct GA_session* session, const GA_json* details, struct
                      GA_auth_handler** call)
```

Get the user's credentials.

Parameters

- **session** – The session to use.
- **details** – The *Get credentials JSON* to get the credentials.
- **call** – Destination for the resulting GA_auth_handler to get the user's credentials. The call handlers result is *Login credentials JSON*. 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_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 wallet 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 current wallet 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_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 *Auth handler 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. The auth_handler object should be destroyed using `GA_destroy_auth_handler` after receiving this status.
- "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 using `GA_destroy_auth_handler` 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": A twofactor code from the "request_code" step, or data from a hardware device is required. If the status JSON contains *Required Data JSON*, then see *GDK Hardware Wallet Interface* for details. Otherwise, to resolve a twofactor 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 hardware 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, or the serialised JSON response for hardware interaction (see *GDK Hardware Wallet Interface*).

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.

Returns the "twofactor_reset" portion of *Two-Factor config JSON* in the GA_auth_handler result.

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_undo_reset** (struct GA_session* *session*, const char* *email*, struct GA_auth_handler** *call*)

Undo a request to begin the two factor authentication reset process.

Returns the "twofactor_reset" portion of *Two-Factor config JSON* in the GA_auth_handler result.

Parameters

- **session** – The session to use.
- **email** – The email address to cancel the reset request for. Must be the email previously passed to *GA_twofactor_reset*.
- **call** – Destination for the resulting GA_auth_handler to request the reset. Returned GA_auth_handler should be freed using *GA_destroy_auth_handler*.

Note: Unlike *GA_twofactor_cancel_reset*, this call only removes the reset request associated with the given email. If other emails have requested a reset, the wallet will still remain locked following this call.

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 two factor reset requests and unlock the wallet for normal operation.

This call requires authentication using an existing wallet twofactor method.

Returns the "twofactor_reset" portion of *Two-Factor config JSON* in the GA_auth_handler result.

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_generate_mnemonic_12** (char** *output*)

Generate a new random 12 word 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 *Network 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

This section describes the various JSON formats used by the library.

2.1 Initialization config JSON

Passed to `GA_init` when initializing the library.

```
{
  "datadir": "/path/to/store/data"
  "tordir": "/path/to/store/tor/data"
  "registrydir": "/path/to/store/registry/data"
  "log_level": "info",
}
```

datadir Mandatory. A directory which gdk will use to store encrypted data relating to sessions.

tordir An optional directory for tor state data, used when the internal tor implementation is enabled in *Connection parameters JSON*. Note that each process using the library at the same time requires its own distinct directory. If not given, a subdirectory "tor" inside "datadir" is used.

registrydir An optional directory for the registry data, used when the network is liquid based. Note that each process using the library at the same time requires its own distinct directory. If not given, a subdirectory "registry" inside "datadir" is used.

log_level Library logging level, one of "debug", "info", "warn", "error", or "none".

2.2 Connection parameters JSON

```
{
  "name": "testnet",
  "proxy": "localhost:9150",
}
```

(continues on next page)

(continued from previous page)

```
"use_tor": true,  
"user_agent": "green_android v2.33",  
"spv_enabled": false,  
"cert_expiry_threshold": 1  
}
```

name The name of the network to connect to. Must match a key from *Network list JSON*.

proxy The proxy connection to pass network traffic through, if any.

use_tor true to enable Tor connections, false otherwise. If enabled and a proxy is not given, a Tor connection will be started internally. If a proxy is given and Tor is enabled, the proxy must support resolving ".onion" domains.

user_agent The user agent string to pass to the server for multisig connections.

spv_enabled true to enable SPV verification for the session, false otherwise.

cert_expiry_threshold Ignore certificates expiring within this many days from today. Used to pre-empt problems with expiring embedded certificates.

2.3 Proxy Settings JSON

Contains the proxy settings in use by a session.

```
{  
  "proxy": "localhost:9150",  
  "use_tor": true  
}
```

proxy The proxy connection being used to pass network traffic through, or an empty string.

use_tor true if Tor is enabled, false otherwise.

2.4 Login credentials JSON

To authenticate with a hardware wallet, pass empty JSON and provide *HW device JSON*.

To authenticate with a mnemonic and optional password:

```
{  
  "mnemonic": "moral lonely ability sail balance simple kid girl inhale master_  
↪dismiss round about aerobic purpose shiver silly happy kitten track kind pattern_  
↪nose noise",  
  "password": ""  
}
```

Or, with a mnemonic and optional BIP39 passphrase:

```
{  
  "mnemonic": "moral lonely ability sail balance simple kid girl inhale master_  
↪dismiss round about aerobic purpose shiver silly happy kitten track kind pattern_  
↪nose noise",  
  "bip39_passphrase": ""  
}
```

To authenticate with a PIN:

```
{
  "pin": "123456",
  "pin_data": {
    "encrypted_data":
    ↪ "0b39c1e90ca6adce9ff35d1780de74b91d46261a7cbf2b8d2fdc21528c068c8e2b26e3bf3f6a2a992e0e1ecfad02203431",
    ↪ "pin_identifier": "38e2f188-b3a8-4d98-a7f9-6c348cb54cfe",
    "salt": "a99/9Qy6P7ON4Umk2FafVQ=="
  }
}
```

pin The PIN entered by the user to unlock the wallet.

pin_data See *PIN data JSON*.

To authenticate a watch-only user:

```
{
  "username": "my_watch_only_username",
  "password": "my_watch_only_password"
}
```

2.5 HW device JSON

Describes the capabilities of an external signing device.

```
{
  "device": {
    "name": "Ledger",
    "supports_ae_protocol": 0,
    "supports_arbitrary_scripts": true,
    "supports_host_unblinding": false,
    "supports_liquid": 1,
    "supports_low_r": false,
  }
}
```

name The unique name of the hardware device.

supports_arbitrary_scripts True if the device can sign non-standard scripts such as CSV.

supports_low_r True if the device can produce low-R ECDSA signatures.

supports_liquid 0 if the device does not support Liquid, 1 otherwise.

supports_host_unblinding True if the device supports returning the Liquid master blinding key.

supports_ae_protocol See “ae_protocol_support_level” enum in the gdk source for details.

The default for any value not provided is false or 0.

2.6 PIN data JSON

Contains the data returned by *GA_encrypt_with_pin*. The caller must persist this data and pass it to *GA_login_user* along with the users PIN in order to allow a PIN login.

```
{
  "encrypted_data":
  ↪ "0b39c1e90ca6adce9ff35d1780de74b91d46261a7cbf2b8d2fdc21528c068c8e2b26e3bf3f6a2a992e0e1ecfad02203431",
  ↪ ",
  "pin_identifier": "38e2f188-b3a8-4d98-a7f9-6c348cb54cfe",
  "salt": "a99/9Qy6P7ON4Umk2FafVQ=="
}
```

2.7 Encrypt with PIN JSON

```
{
  "pin": "...",
  "plaintext": {}
}
```

pin The PIN to protect the server provided key.

plaintext The json to encrypt. For instance it can be the *Login credentials JSON* with the mnemonic.

2.8 Encrypt with PIN Result JSON

```
{
  "pin_data": "...",
}
```

pin_data See *PIN data JSON*.

2.9 Wallet identifier request JSON

Describes the wallet to compute an identifier for using *GA_get_wallet_identifier*. You may pass *Login credentials JSON* to compute an identifier from a mnemonic and optional password, note that PIN or watch-only credentials cannot be used. otherwise, pass the wallets master xpub as follows:

```
{
  "master_xpub":
  ↪ "tpubD8G8MPH9RK9uk4EV97RxxhzaY8SJPuWxnViHUWji92i8B7vYdht797PPDrJveeatHnKxonJe8SbaScAC1YJ8xAzZbH9Uvvy",
  ↪ ",
}
```

master_xpub The base58-encoded BIP32 extended master public key of the wallet.

2.10 Get credentials JSON

Accepts an optional password to encrypt the mnemonic.

```
{
  "password": ""
}
```

2.11 Subaccount JSON

Describes a subaccount within the users wallet. Returned by *GA_get_subaccount* and as the array elements of *GA_get_subaccounts*.

```
{
  "hidden": false,
  "name": "Subaccount Name",
  "pointer": 0,
  "receiving_id": "GA7ZnuhsieSMNp2XAB3oEyLy75peM",
  "recovery_chain_code": "",
  "recovery_pub_key": "",
  "recovery_xpub": "",
  "required_ca": 0,
  "type": "2of2"
  "bip44_discovered": false
}
```

hidden Whether the subaccount is hidden.

name The name of the subaccount.

pointer The subaccount number.

receiving_id The Green receiving ID for the subaccount.

recovery_chain_code For "2of3" subaccounts, the BIP32 chaincode of the users recovery key.

recovery_pub_key For "2of3" subaccounts, the BIP32 public key of the users recovery key.

recovery_xpub For "2of3" subaccounts, the BIP32 xpub of the users recovery key.

required_ca For "2of2_no_recovery" subaccounts, the number of confidential addresses that the user must upload to the server before transacting.

type For multisig subaccounts, one of "2of2", "2of3" or "2of2_no_recovery". For singlesig subaccounts, one of "p2pkh", "p2wpkh" or "p2sh-p2wpkh".

bip44_discovered Singlesig only. Whether or not this subaccount contains at least one transaction.

2.12 Subaccount update JSON

Describes updates to be made to a subaccount via *GA_update_subaccount*.

```
{
  "hidden": true,
  "name": "New name",
  "subaccount": 1
}
```

hidden If present, updates whether the subaccount will be marked hidden.

name If present, updates the name of the subaccount.

subaccount The subaccount to update.

2.13 Subaccounts list JSON

```
{
  "subaccounts": [
    { },
    { }
  ]
}
```

subaccounts An array of *Subaccount JSON* elements for each of the users subaccounts.

2.14 Transaction list JSON

```
[
  {
    "addressees": [
      ""
    ],
    "block_height": 0,
    "calculated_fee_rate": 1004,
    "can_cpfp": true,
    "can_rbf": false,
    "created_at_ts": 1551280324000000,
    "fee": 206,
    "fee_rate": 1004,
    "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
      }
    ],
    "memo": "",
    "outputs": [
      {
        "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,

```

(continues on next page)

(continued from previous page)

```

        "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_size": 370,
"transaction_vsize": 205,
"transaction_weight": 820,
"txhash": "fe50531d94fae597d9e209582a401e62b1f705ace93eca94fe2e42f187456e4a",
"type": "incoming",
"user_signed": true,
"vsize": 205,
"spv_verified": "disabled"
}
]

```

2.15 Transaction details JSON

Contains information about a transaction that may not be associated with the users wallet. Returned by `GA_get_transaction_details`.

```

{
  "transaction":
  ↪ "020000000000101ab0dec345ed48b0761411306eae50f90dd34f3c8598e48f1c3ad324a862bc72b0000000000feffffff02
  ↪ ",
  "transaction_locktime": 432,
  "transaction_size": 223,
  "transaction_version": 2,
  "transaction_vsize": 142,
  "transaction_weight": 565,
  "txhash": "dc5c908a6c979211e6482766adb69cbcb760c92923671f6304d12a3f462a2b0"
}

```

2.16 Create transaction JSON

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

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

2.17 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":
    ↪ "522102da0e5f74219dadbd392dc3157c43c3636e237005e7f3976a338e519901fdf9e32103326c44e51893994677bb43e
    ↪ ",
    "script_type": 14,
    "service_xpub":
    ↪ "tpubEAUTpVqYYmDxumXSPwZEgCRC5HZXagbsATdv3wUMweyDrJY4fVDt89ogtpBxa9ynpXB3AyGen3Ko4S8ewpWkkvQsvYP86
    ↪ ",
    "subaccount": 0,
    "subtype": null,
    "user_path": [
      1,
      492
    ]
  ]
}
```

(continues on next page)

(continued from previous page)

```

},
"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": {
  "btc": 100000
},
"send_all": false,
"server_signed": false,
"subaccount": 0,
"transaction":
↪ "020000000000101c01365291a12d995d7afc3234f4e86d3e064f175ab5a7d47e631de7f293a930901000000230000000000
↪ ",
"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":
↪ "tpubEATUpVqYYmDxumXSPwZEgCRC5HZXagbsATdv3wUMweyDrJY4fVDt89ogtpBxa9ynpXB3AyGen3Ko4S8ewpWkkvQsvYP86
↪ ",
    "subaccount": 0,
    "subtype": null,
    "user_path": [
      1,
      492
    ]
  },
  {
    "address": "2MtcMpWnde3tf5vfnHXKBaWuAUS8j89771",
    "is_change": false,
    "satoshi": 100000,
    "script": "a9140ef7660003133f69023f0436dc8bcf427941dcf587"
  }
],
"transaction_size": 372,
"transaction_version": 2,
"transaction_vsize": 206,
"transaction_weight": 822,
"used_utxos": [
  0
],
"user_signed": false,
"utxo_strategy": "default",

```

(continues on next page)

(continued from previous page)

```

"utxos": [
  {
    "address_type": "p2wsh",
    "block_height": 1448369,
    "pointer": 475,
    "prevout_script":
    ↪ "522103bad7ac76143368781c4ac3e7afbb63cd6b52f2a923c715576804aa1046cabcl1a210264f5fa70969907861ebdb2b2
    ↪ ",
    "pt_idx": 1,
    "satoshi": 4989794,
    "script_type": 14,
    "sequence": 4294967293,
    "service_xpub":
    ↪ "tpubEATpVqYYmDxumXSPwZEgCRC5HZXagbsATdv3wUMweyDrJY4fVDt89ogtpBxa9ynpXB3AyGen3Ko4S8ewpWkkvQsvYP86
    ↪ ",
    "subaccount": 0,
    "subtype": 0,
    "txhash": "09933a297fde31e6477d5aab75f164e0d3864e4f23c3afd795d9121a296513c0",
    "user_path": [
      1,
      475
    ]
  },
  {
    "address_type": "p2wsh",
    "block_height": 1448369,
    "pointer": 474,
    "pt_idx": 0,
    "satoshi": 10000,
    "script_type": 14,
    "subaccount": 0,
    "subtype": 0,
    "txhash": "09933a297fde31e6477d5aab75f164e0d3864e4f23c3afd795d9121a296513c0"
  }
],
"memo": ""
}

```

2.18 Send transaction JSON

```

{
  "addressees": [
    {
      "address": "2NDwUefHRbbHuGsumAWMbRZUzigrtBYkwrq",
      "bip21-params": null,
      "satoshi": 100000
    }
  ],
  "addressees_read_only": false,
  "amount_read_only": false,
  "available_total": 4999588,
  "calculated_fee_rate": 1281,
  "change_address": {
    "address": "2Mtpg961bP6WH9cQvY2qS4rnuceorBrnutn",

```

(continues on next page)

(continued from previous page)

```

    "address_type": "p2wsh",
    "branch": 1,
    "pointer": 497,
    "script":
    ↪ "52210350683b20cc33983f818c9b50606909622dbc4387a17699e5ae09b9d5d1b3111c21028598a36a99fbda64ff1d942a
    ↪ ",
    "script_type": 14,
    "service_xpub":
    ↪ "tpubEATpVqYYmDxumXSPwZEgCRC5HZXagbsATdv3wUMweyDrJY4fVDt89ogtpBxa9ynpXB3AyGen3Ko4S8ewpWkkvQsvYP86
    ↪ ",
    "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":
    ↪ "0200000000001027ff3490a29a2fe73f07e3d3f8740249d61c0025fdc0819586dd9443bc6a00bd30100000023220020ed1
    ↪ ",
"transaction_locktime": 1483342,
"transaction_outputs": [
    {
        "address": "2NDwUefHRbbHuGsumAWMbRZUzigrtBYkwrq",
        "is_change": false,
        "satoshi": 100000,
        "script": "a914e2ff64a1ca976947d47b6b2d214af96d5942e1b287"
    },
    {
        "address": "2Mtpg961bP6WH9cQvY2qS4rnuceoRBrnutn",
        "address_type": "p2wsh",
        "branch": 1,
        "is_change": true,
        "pointer": 497,
        "satoshi": 109663,
        "script": "a9141114baed477ca8fb65f856b96f860acc52619a6fc87",
        "script_type": 14,
        "service_xpub":
        ↪ "tpubEATpVqYYmDxumXSPwZEgCRC5HZXagbsATdv3wUMweyDrJY4fVDt89ogtpBxa9ynpXB3AyGen3Ko4S8ewpWkkvQsvYP86
        ↪ ",
        "subaccount": 0,
        "subtype": null,

```

(continues on next page)

(continued from previous page)

```

    "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,
        "pointer": 474,
        "prevout_script":
→ "522102ff54a17dc6efe168673dbf679fe97e06b5cdcaf7dea8ab83dc6732350cd1b4e4210279979574e0743b4659093c0
→ ",
        "pt_idx": 0,
        "satoshi": 10000,
        "script_type": 14,
        "sequence": 4294967293,
        "service_xpub":
→ "tpubEATpVqYYmDxumXSPwZEgCRC5HZXagbsATdv3wUMweyDrJY4fVDt89ogtpBxa9ynpXB3AyGen3Ko4S8ewpWkkvQsvYP86
→ ",
        "subaccount": 0,
        "subtype": 0,
        "txhash": "09933a297fde31e6477d5aab75f164e0d3864e4f23c3afd795d9121a296513c0",
        "user_path": [
            1,
            474
        ]
    },
    {
        "address_type": "p2wsh",
        "block_height": 0,
        "pointer": 493,
        "prevout_script":
→ "522102c9465e8b6e98848428b90f21291a19c62fcb20d2dbff76217068219cada5f7a921022e831b15a4faa339ed9a09a
→ ",
        "pt_idx": 1,
        "satoshi": 200000,
        "script_type": 14,
        "sequence": 4294967293,
        "service_xpub":
→ "tpubEATpVqYYmDxumXSPwZEgCRC5HZXagbsATdv3wUMweyDrJY4fVDt89ogtpBxa9ynpXB3AyGen3Ko4S8ewpWkkvQsvYP86
→ ",
        "subaccount": 0,
        "subtype": 0,
        "txhash": "d30ba0c63b44d96d581908dc5f02c0619d2440873f3d7ef073fea2290a49f37f",
        "user_path": [

```

(continues on next page)

(continued from previous page)

```

    1,
    493
  ]
},
{
  "address_type": "p2wsh",
  "block_height": 0,
  "pointer": 494,
  "pt_idx": 0,
  "satoshi": 4789588,
  "script_type": 14,
  "subaccount": 0,
  "subtype": 0,
  "txhash": "d30ba0c63b44d96d581908dc5f02c0619d2440873f3d7ef073fea2290a49f37f"
}
]
}

```

2.19 Sign PSBT JSON

```

{
  "psbt": "...",
  "utxos": [],
  "blinding_nonces": [],
}

```

psbt The PSBT or PSET encoded in base64 format.

utxos The UTXOs that should be signed, in the format returned by *GA_get_unspent_outputs*. UTXOs that are not inputs of the PSBT/PSET can be included. Caller can avoid signing an input by not passing in its UTXO.

blinding_nonces For "2of2_no_recovery" subaccounts only, the blinding nonces in hex format for all outputs.

2.20 Sign PSBT Result JSON

```

{
  "psbt": "...",
  "utxos": [],
}

```

psbt The input PSBT or PSET in base64 format, with signatures added for all inputs signed.

utxos The UTXOs corresponding to each signed input, in the order they appear in the PSBT transaction.

2.21 PSBT Get Details JSON

```
{
  "psbt": "...",
  "utxos": [],
}
```

psbt The PSBT or PSET encoded in base64 format.

utxos The UTXOs owned by the wallet, in the format returned by *GA_get_unspent_outputs*. UTXOs that are not inputs of the PSBT/PSET can be included.

2.22 PSBT Get Details Result JSON

```
{
  "inputs": [
    {
      "asset_id": "...",
      "satoshi": 0,
      "subaccount": 0,
    },
  ],
  "outputs": [
    {
      "asset_id": "...",
      "satoshi": 0,
      "subaccount": 0,
    },
  ],
}
```

Note: Inputs and outputs might have additional fields that might be removed or changed in following releases.

2.23 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, 1499] }
```

2.24 Two-Factor config JSON

Describes the wallets enabled two factor methods, current spending limits, and two factor reset status.

```
{
  "all_methods": [
    "email",
    "sms",
    "phone",
    "gauth"
  ],
  "any_enabled": true,
}
```

(continues on next page)

(continued from previous page)

```

"email": {
  "confirmed": true,
  "data": "***@g***",
  "enabled": true
},
"enabled_methods": [
  "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
}
}

```

twofactor_reset/days_remaining The number of days remaining before the wallets two factor authentication is reset, or -1 if no reset procedure is underway.

twofactor_reset/is_active Whether or not the wallet is currently undergoing the two factor reset procedure.

twofactor_reset/is_disputed Whether or not the wallet two factor reset procedure is disputed.

2.25 Settings JSON

```

{
  "altimeout": 10,
  "csvtime": 51840,

```

(continues on next page)

(continued from previous page)

```

"nlocktime": 12960,
"notifications": {
  "email_incoming": true,
  "email_outgoing": true
},
"pgp": "",
"pricing": {
  "currency": "EUR",
  "exchange": "KRAKEN"
},
"required_num_blocks": 12,
"sound": true,
"unit": "BTC"
}

```

2.26 Receive address details JSON

```

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

```

address The wallet address in base58, bech32 or blech32 encoding.

address_type One of "csv", "p2sh", "p2wsh" (multisig), or "p2pkh", "p2sh-p2wpkh", "p2wpkh" (singlesig), indicating the type of address.

branch Always 1, used in the address derivation path for subaccounts.

pointer The address number/final number in the address derivation path.

script The scriptpubkey of the address.

script_type Integer representing the type of script.

subaccount The subaccount this address belongs to. Matches "pointer" from *Subaccounts list JSON* or *Subaccount JSON*.

subtype For "address_type" "csv", the number of CSV blocks referenced in "script", otherwise, 0.

user_path The BIP32 path for the user key.

For Liquid addresses, the following additional fields are returned:

```

{
  "blinding_key": "02a519491b130082a1abbe17395213b46dae43c3e1c05b7a3dbd2157bd83e88a6e
  ↪ ",

```

(continues on next page)

(continued from previous page)

```

"blinding_script": "a914c2427b28b2796243e1e8ee65be7598d465264b0187",
"is_blinded": true,
"unblinded_address": "XV4PaYgbaJdPnYaJDzE41TpbBF6yBieeyd"
}

```

blinding_key The blinding key used to blind this address.

blinding_script The script used to generate the blinding key via <https://github.com/satoshilabs/slips/blob/master/slip-0077.md>.

is_blinded Always `true`.

unblinded_address The unblinded address. This is provided for informational purposes only and should not be used to receive.

2.27 Previous addresses request JSON

Contains the query parameters for requesting previously generated addresses using `GA_get_previous_addresses`.

```

{
  "subaccount": 0,
  "last_pointer": 0,
}

```

subaccount The value of “pointer” from *Subaccounts list JSON* or *Subaccount JSON* for the subaccount to fetch addresses for. Default 0.

last_pointer The address pointer from which results should be returned. If this key is not present, the newest generated addresses are returned. If present, the “last_pointer” value from the resulting *Previous addresses JSON* should then be given, until sufficient pages have been fetched or the “last_pointer” key is not present indicating all addresses have been fetched.

is_internal Singlesig only. Whether or not the user key belongs to the internal chain.

2.28 Previous addresses JSON

Contains a page of previously generated addresses, from newest to oldest.

```

{
  "last_pointer": 2,
  "list": [
    {
      "address": "2N52RVsChsCi439PpJlHn8fHCiTrRjcAEiL",
      "address_type": "csv",
      "branch": 1,
      "is_internal": false,
      "pointer": 2,
      "script":
↪ "2102df992d7fa8f012d61048349e366f710aa0168a1c08606d7bebb65f980ccf2616ad2102a503dfc70ad1f1a510f7e3c"
↪ ,
      "script_type": 15,
      "subaccount": 0,
    }
  ]
}

```

(continues on next page)

(continued from previous page)

```

    "subtype": 51840,
    "tx_count": 0,
    "user_path": [
        1,
        2
    ],
  },
  {
    "address": "2MzyxeSfodsJkj4YYAyyNpGwqpvdze7qLSf",
    "address_type": "csv",
    "branch": 1,
    "is_internal": false,
    "pointer": 1,
    "script":
    ↪ "2102815c7ba597b1e0f08357ddb346dab3952b2a76e189efc9ebde51ec005df0b41cad210328154df2714de6b15e74033",
    ↪ ",
    "script_type": 15,
    "subaccount": 0,
    "subtype": 51840,
    "tx_count": 0,
    "user_path": [
        1,
        1
    ],
  }
],
}

```

last_pointer If present indicates that there are more addresses to be fetched, and the caller to get the next page should call again *GA_get_previous_addresses* passing this value in *Previous addresses request JSON*. If not present there are no more addresses to fetch.

list Contains the current page of addresses in *Receive address details JSON* format.

2.29 Unspent outputs request JSON

Describes which unspent outputs to return from *GA_get_unspent_outputs*, or which unspent outputs to include in the balance returned by *GA_get_balance*.

```

{
  "subaccount": 3,
  "num_confs": 0,
  "all_coins": false,
  "expired_at": 99999,
  "confidential": false,
  "dust_limit": 546
}

```

subaccount The subaccount to fetch unspent outputs for.

num_confs Pass 0 for unconfirmed UTXOs or 1 for confirmed.

all_coins Pass `true` to include UTXOs with status `frozen`. Defaults to `false`.

expired_at If given, only UTXOs where two-factor authentication expires by the given block are returned.

confidential Pass `true` to include only confidential UTXOs. Defaults to `false`.

dust_limit If given, only UTXOs with a value greater than the limit value are returned.

2.30 Unspent outputs JSON

Contains the filtered unspent outputs.

```
{
  "unspent_outputs": {
    "btc": [
      {
        "txhash": "09933a297fde31e6477d5aab75f164e0d3864e4f23c3afd795d9121a296513c0",
        "pt_idx": 0,
        "satoshi": 10000,
        "block_height": 1448369,
        "address_type": "p2wsh",
        "is_internal": false,
        "pointer": 474,
        "subaccount": 0,
        "prevout_script":
        ↪ "522102ff54a17dc6efe168673dbf679fe97e06b5cdcaf7dea8ab83dc6732350cd1b4e4210279979574e0743b4659093c00",
        ↪ ",
        "user_path": [
          1,
          474
        ],
        "public_key":
        ↪ "0279979574e0743b4659093c005256c812f68f512c50d7d1622650b891de2cd61e",
        "expiry_height": 1458369,
        "script_type": 14,
        "user_status": 0,
        "subtype": 0,
      },
    ],
  }
}
```

txhash The txid of the transaction.

pt_idx The index of the output, the vout.

satoshi The amount of the output.

block_height The height of the block where the transaction is included. Is 0 if the transaction is unconfirmed.

address_type One of "csv", "p2sh", "p2wsh" (multisig), or "p2pkh", "p2sh-p2wpkh", "p2wpkh" (singlesig), indicating the type of address.

is_internal Whether or not the user key belongs to the internal chain.

pointer The user key number/final number in the derivation path.

subaccount The subaccount this output belongs to. Matches "pointer" from *Subaccounts list JSON* or *Subaccount JSON*.

prevout_script The script being signed, the script code.

user_path The BIP32 path for the user key.

public_key Singlesig only. The user public key.

expiry_height Multisig only. The block height when two-factor authentication expires.

script_type Multisig only. Integer representing the type of script.

user_status Multisig only. 0 for "default" and 1 for "frozen".

subtype Multisig only. For "address_type" "csv", the number of CSV blocks referenced in "script", otherwise, 0.

For Liquid instead of having the "btc" field, there are (possibly) multiple fields, one for each asset owned, and the keys are the hex-encoded policy ids.

For Liquid the inner maps have additional fields:

```
{
  "confidential": true,
  "asset_id": "e4b76d990f27bf6063cb66ff5bbc783d03258a0406ba8ac09abab7610d547e72",
  "assetblinder": "aedb6c37d0ea0bc64fbc7036b52d0a0784da0b1ca90ac918c19ee1025b0c944c",
  "amountblinder": "3be117b88ba8284b05b89998bdee1ded8cd5b561ae3d05bcd91d4e8abab2cd47",
  "asset_tag": "0b103a2d34cf469987dd06937919f9dae8c9856be17c554fd408fdc226b1769e59",
  "commitment": "094c3f83d5bac22b527ccac141fe04883d79bf04aef10a1dd42f501c5b51318907",
  "nonce_commitment":
  ↪ "0211b39afe463473e428cfafd387f9c85b350f440131fad03aa5f4809b6c834f30",
}
```

confidential Whether or not the output is confidential.

asset_id The hex-encoded asset id.

assetblinder The hex-encoded asset blinder (asset blinding factor, abf)

amountblinder The hex-encoded amount blinder (value blinding factor, vbf)

asset_tag The hex-encoded asset commitment.

commitment The hex-encoded value commitment.

nonce_commitment The hex-encoded nonce commitment.

2.31 Unspent outputs set status JSON

Valid status values are "default" for normal behaviour or "frozen". Frozen outputs are hidden from the caller's balance and unspent output requests, are not returned in nlocktime emails, and cannot be spent. An account containing frozen outputs can be deleted, whereas an account with unfrozen outputs can not.

Freezing an output requires two factor authentication. Outputs should only be frozen in response to e.g. a dust attack on the wallet. Once a wallet is deleted, any frozen outputs it contained will be unspendable forever.

Note: Only outputs of value less than two times the dust limit can be frozen.

```
{
  "list" : [
    {
      "txhash": "09933a297fde31e6477d5aab75f164e0d3864e4f23c3afd795d9121a296513c0",
      "pt_idx": 1,
      "user_status": "frozen"
    }
  ]
}
```

(continues on next page)

(continued from previous page)

```

    }
  ]
}
```

2.32 Transactions details JSON

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

2.33 Network JSON

```
{
  "address_explorer_url": "",
  "bech32_prefix": "bcrt",
  "bip21_prefix": "bitcoin",
  "csv_buckets": [
    144,
    4320,
    51840
  ],
  "development": true,
  "electrum_tls": false,
  "electrum_url": "localhost:19002",
  "liquid": false,
  "mainnet": false,
  "name": "Localtest",
  "network": "localtest",
  "p2pkh_version": 111,
  "p2sh_version": 196,
  "server_type": "green",
  "service_chain_code":
  ↪ "b60befcc619bb1c212732770fe181f2f1aa824ab89f8aab49f2e13e3a56f0f04",
  "service_pubkey":
  ↪ "036307e560072ed6ce0aa5465534fb5c258a2ccfbc257f369e8e7a181b16d897b3",
  "spv_multi": false,
  "spv_servers": [],
  "spv_enabled": false,
  "tx_explorer_url": "",
  "wamp_cert_pins": [],
  "wamp_cert_roots": [],
  "wamp_onion_url": "",
  "wamp_url": "ws://localhost:8080/v2/ws"
}
```

2.34 Network list JSON

Contains a list of all available networks the API can connect to.

```
{
  "all_networks": [
    "mainnet",
    "liquid",
    "testnet"
  ],
  "liquid": { },
  "mainnet": { },
  "testnet": { },
}
```

For each network listed, a *Network JSON* element is present containing the networks information.

2.35 Transaction limits JSON

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

2.36 Two-factor detail JSON

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

2.37 Auth handler status JSON

Describes the status of a `GA_auth_handler`. Returned by `GA_auth_handler_get_status`.

The data returned depends on the current state of the handler, as follows:

- "done":

```
{
  "status": "done",
  "action": "disable_2fa",
  "result": {}
}
```

action The action being processed.

result The data returned from the call, if any.

- "error":

```
{
  "status": "error",
  "action": "disable_2fa",
  "error": "Incorrect code"
}
```

action The action being processed.

error A text description of the error that occurred.

- "call":

```
{
  "status": "call",
  "action": "disable_2fa"
}
```

action The action being processed.

- "request_code":

```
{
  "status": "request_code",
  "action": "disable_2fa",
  "methods": [ "email", "sms", "phone", "gauth", "telegram" ]
}
```

action The action being processed.

methods A list of the two factor methods the user has enabled.

- "resolve_code" (two factor):

```
{
  "status": "resolve_code",
  "action": "disable_2fa",
  "method": "email",
  "auth_data": {},
  "attempts_remaining": "3"
}
```

action The action being processed.

method The two factor method the user should fetch the code to enter from.

auth_data Method-specific ancillary data for resolving the call.

attempts_remaining If present, the number of incorrect attempts that can be made before the call fails.

- "resolve_code" (hardware wallet/external device):

```
{
  "status": "resolve_code",
  "action": "disable_2fa",
  "required_data": {
    "action": "get_xpubs",
    "device": {}
  }
}
```

action The action being processed.

required_data Contains the data the HWW must provide, see *GDK Hardware Wallet Interface*.

2.38 Reconnect JSON

Controls session and internal Tor instance reconnection behaviour.

```
{
  "hint": "connect",
  "tor_hint": "connect"
}
```

hint Optional, must be either "connect" or "disconnect" if given.

tor_hint Optional, must be either "connect" or "disconnect" if given.

For both hint types, "disconnect" will disconnect the underlying network connection used by the session, while "connect" will reconnect it. If a hint is not given, no action will be taken for that connection type.

Each session will automatically attempt to reconnect in the background when they detect a disconnection, unless "disconnect" is passed to close the connection first. The session will be notified using a *Network notification* when the underlying network connection changes state.

For environments such as mobile devices where networking may become unavailable to the callers application, the network must be disconnected and reconnected using *GA_reconnect_hint* in order for connectivity to be resumed successfully. In particular, when using the built-in Tor implementation to connect, failure to do so may result in Tor failing to connect for the remaining lifetime of the application (this is a Tor limitation).

2.39 Convert amount JSON

Amounts to convert are passed with a single key containing the unit value to convert, returning all possible conversion values for that value. See *Amount JSON* for the list of unit values available.

For example, to convert satoshi into all available units:

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

If "fiat_currency" and "fiat_rate" members are provided, the fiat conversion will fall back on these values if no fiat rates are available. Callers can check the "is_current" member in the result *Amount JSON* to determine if the fall back values were used.

For example, to convert bits into all available units, with a fiat conversion fallback:

```
{
  "bits": "20344.69",
  "fiat_currency": "USD",
  "fiat_rate": "42161.22"
}
```

2.40 Amount JSON

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

(continues on next page)

(continued from previous page)

```

"sats": "2034469",
"subaccount": 0,
"ubtc": "20344.69"
"is_current": true
}

```

fiat_currency Set to the users fiat currency if available, otherwise an empty string.

fiat_rate Set to the users fiat exchange rate if available, otherwise null.

is_current true if the "fiat_currency" and "fiat_rate" members are current.

2.41 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"], "BTC AVG": [], "BTC CHINA": [],
  ↪ "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"]}
}

```

2.42 HTTP parameters JSON

```

{
  "accept": "json"
  "method": "GET"
  "urls": [
    "https://assets.blockstream.info/index.json"
    "http://vi5flmr4z3h3luup.onion/index.json"
  ]
  "proxy": "localhost:9150"
  "headers": { "If-Modified-Since": "Mon, 02 Sep 2019 22:39:39 GMT" }
  "timeout": 10
}

```

2.43 Locktime details JSON

```

{
  "value": 65535
}

```

2.44 Asset parameters JSON

```
{
  "assets": true,
  "icons": true,
  "refresh": true
}
```

2.45 Get assets parameters JSON

```
{
  "assets_id": [ "6f0279e9ed041c3d710a9f57d0c02928416460c4b722ae3457a11eec381c526d",
  ↪ "144c654344aa716d6f3abcc1ca90e5641e4e2a7f633bc09fe3baf64585819a49" ],
}
```

2.46 Asset informations JSON

```
{
  "assets": {
    "6f0279e9ed041c3d710a9f57d0c02928416460c4b722ae3457a11eec381c526d": {
      "asset_id": "6f0279e9ed041c3d710a9f57d0c02928416460c4b722ae3457a11eec381c526d
  ↪",
      "contract": null,
      "entity": null,
      "issuance_prevout": {
        "txid": "0000000000000000000000000000000000000000000000000000000000000000
  ↪",
        "vout": 0
      },
      "issuance_txin": {
        "txid": "0000000000000000000000000000000000000000000000000000000000000000
  ↪",
        "vin": 0
      },
      "issuer_pubkey": "",
      "name": "btc",
      "precision": 8,
      "ticker": "L-BTC",
      "version": 0
    }
  },
  "icons": {
    "6f0279e9ed041c3d710a9f57d0c02928416460c4b722ae3457a11eec381c526d": "BASE64"
  }
}
```

2.47 Error details JSON

```
{
  "details": "assertion failure: ../src/ga_session.cpp:rename_subaccount:2166:Unknown_
↪subaccount"
}
```

2.48 Get Subaccounts parameters JSON

Parameters controlling the *GA_get_subaccounts* call.

```
{
  "refresh": false
}
```

refresh If set to `true`, subaccounts are re-discovered if appropriate for the session type. Note that this will take significantly more time if set. Defaults to `false`.

This section describes the notifications emitted by the library.

All notifications contain an "event" element which describes the type of event being notified. The notification data is available under an element named with the content of the "event" element.

3.1 Network notification

Notified when the state of a session's underlying network connection changes.

```
{
  "event": "network",
  "network": {
    "wait_ms": 1000,
    "current_state": "disconnected",
    "next_state": "connected"
  }
}
```

current_state One of "connected" or "disconnected". The current state of the network connection.

next_state One of "connected" or "disconnected". The next state that the connection will move to. If this value is the same as "current_state" then no state change is currently in progress.

wait_ms The number of milliseconds before the current state will change to the next state. 0 if the change will happen immediately or no change is due to occur.

3.2 Tor notification

Notified when using the built-in Tor connection during connection establishment.

```
{
  "event": "tor",
  "tor": {
    "progress": 20,
    "summary": "Establishing an encrypted directory connection",
    "tag": "onehop_create"
  }
}
```

tor/progress An integer from 0-100 indicating the progress percentage.

tor/summary A human-readable summary of the current progress step.

tor/onehop_create A fixed identifier string for the current progress step.

3.3 Settings notification

Notified upon successful authentication. Describes the current wallet settings.

```
{
  "event": "settings",
  "settings": {}
}
```

settings Contains the *Settings JSON* of the user.

3.4 Two factor reset notification

Notified by multisig sessions upon successful authentication. Describes the current two factor reset status of the wallet.

```
{
  "event": "twofactor_reset",
  "twofactor_reset": {}
}
```

twofactor_reset Contains the "twofactor_reset" portion of *Two-Factor config JSON*.

3.5 Block notification

Notified when a new block is mined by the network.

```
{
  "event": "block",
  "block": {
    "block_hash": "00000000a09b62cc7c076cf8bb25840e67bb5f9f47492f8a82a09105a6aab72d",
    "block_height": 2138311,
    "initial_timestamp": 1489943482,
    "previous_hash":
    ↪ "00000000000000bcbf344da3c3d691f5581136bf78c52de4c712949541f0ccf3c"
  }
}
```

block/block_hash The hash of the block.

block/block_height The height of the block.

block/initial_timestamp Multisig only. The time that the users wallet was created, in seconds since the epoch.

block/previous_hash The hash of the block prior to this block.

3.6 Transaction notification

Notified when a new transaction is received by the wallet.

```
{
  "event": "transaction",
  "transaction": {
    "satoshi": 50000,
    "subaccounts": [
      0
    ],
    "txhash": "2bee55e07ab6cc520487f57cb74e87c2960d5f01d291d34f6b395417a276a42c",
    "type": "incoming"
  }
}
```

transaction/satoshi Bitcoin only. The net amount of the transaction (always positive).

transaction/subaccounts The wallet subaccounts the transaction affects.

transaction/txhash The txid of the transaction.

transaction/type Bitcoin only. One of "incoming", "outgoing" or "redeposit".

3.7 Ticker notification

Notified when the user's exchange rate changes.

```
{
  "event": "ticker",
  "ticker": {
    "currency": "NZD",
    "exchange": "KIWICOIN",
    "rate": "44100.84"
  }
}
```

ticker/currency The user's chosen fiat currency.

ticker/exchange The user's chosen exchange source.

ticker/rate The price of 1 Bitcoin expressed in the user's fiat currency, expressed as a floating point string.

GDK Hardware Wallet Interface

This section details the format of data requests from hardware wallet interaction during resolution of `GA_auth_handler` processing when `GA_auth_handler_get_status` returns the status `"resolve_code"` with a `"required_data"` element present.

4.1 Required Data JSON

Returned as an element `"required_data"` of *Auth handler status JSON* when data is required from a registered hardware device.

```
{
  "action": "",
  "device": {
  },
}
```

action Describes the hardware wallet data requested.

device Contains the *HW device JSON* originally registered with the session.

Additional fields will be present depending on the action requested, as follows:

4.2 Hardware Get XPubs Action

When `"action"` is `"get_xpubs"`, this describes a request to compute one or more xpubs from the wallet's master key.

```
{
  "paths": [ [], [ 2147501889 ] ]
}
```

paths An array of unsigned 32-bit integer arrays representing each xpub to fetch. The integer values should be interpreted per BIP32, i.e. the topmost bit may be set to indicate a private derivation in the path. An empty array indicates that the top level xpub should be returned.

Expected response:

```
{
  "xpubs": [
    ↪ "tpubD8G8MPH9RK9uk4EV97RxhzaY8SJPuWxNViHUWji92i8B7vYdht797PPDrJveethnKxonJe8SbaScAC1YJ8xAzZbH9Uvy
    ↪ ",
    ↪ "tpubD6NzVbkrYhZ4X9jwmpJxg1kjEJTQgkrnHNEWww2e86X1eUfWu1f7hZpgezAyWUk5zRt4fMPHB33CXrvJSYHHAoVMFXrfx
    ↪ "
  ]
}
```

xpubs An array of base58-encoded BIP32 extended public keys, in the same order as the "paths" elements in the request.

4.3 Hardware Get Master Blinding Key Action

When "action" is "get_master_blinding_key", this describes a request to return the wallet's SLIP0077 master blinding key if the user allows this.

Note: This action is only returned when using the Liquid network.

No request data is currently associated with this request.

Expected response:

```
{
  "master_blinding_key":
  ↪ "512cd6c0b73452a2414e9d86d37cdcc8283b44f0b6dd2b1eec23c59ff12b4f7e5949569b3430220dafce1e0e299a2a6f3
  ↪ "
}
```

master_blinding_key The 512-bit or 256-bit master blinding key for the wallet, hex-encoded. If a 256-bit key is returned, it should be the lower 256-bits of the SLIP0021 derived node as specified in <https://github.com/satoshilabs/slips/blob/master/slip-0077.md>.

Note: If the user denies the request to share the key, an empty string should be returned.

4.4 Hardware Sign Message Action

When "action" is "sign_message", this describes a request to sign a message using the given path.

```
{
  "message": "A text message to sign",
  "path": [ 1195487518 ],
```

(continues on next page)

(continued from previous page)

```
"use_ae_protocol": false
}
```

message The message to be utf-8 encoded and signed.

path The path from the wallet's master key to the key that the message should be signed with.

use_ae_protocol `true` if the hardware device advertises Anti-Exfil support and it should be used for signing, `false` otherwise.

Expected response:

```
{
  "signature":
  ↪ "304402207c673ef4255873cf095016c98c4982cea9a5133060b66a380f1bf3880e54f6c8022056fd731cbdd44cd9636621
  ↪ "
}
```

signature The hex-encoded ECDSA signature in DER encoding corresponding to the given message.

4.5 Hardware Get Blinding Public Keys Action

When "action" is "get_blinding_public_keys", this describes a request to compute blinding public keys from wallet scripts.

Note: This action is only returned when using the Liquid network.

```
{
  "scripts": [ "a91403f650e2434916d5b7f124de8f673442b696282887" ]
}
```

scripts An array of hex-encoded scripts for which a blinding key should be generated.

Expected response:

```
{
  "public_keys": [ "02045e92b8f68bd066180c05a39969f862a67f4efc8f5d7aeca32c627a463b8f27
  ↪ " ]
}
```

public_keys An array of hex-encoded compressed public keys for blinding the given scripts.

4.6 Hardware Get Blinding Nonces Action

When "action" is "get_blinding_nonces", this describes a request to compute blinding nonces and possibly blinding public keys for the given scripts and shared public keys.

Note: This action is only returned when using the Liquid network.

```
{
  "blinding_keys_required": true
  "scripts": [ "a91403f650e2434916d5b7f124de8f673442b696282887" ],
  "public_keys": [ "035f242d49b88ca17948b156263e1f0c86d2cc9e9ff316b058dbbdb351e34bc9aa"
  ↪ " ]
}
```

blinding_keys_required `true` if the blinding public keys must be returned, `false` otherwise. Blinding public keys are not requested if the master blinding key has previously been given.

public_keys An array of hex-encoded compressed shared public keys for computing the nonces.

scripts An array of hex-encoded scripts for which a blinding key should be generated and then the nonce computed using the public key given.

Expected response:

```
{
  "public_keys": [ "02045e92b8f68bd066180c05a39969f862a67f4efc8f5d7aeca32c627a463b8f27"
  ↪ " ]
  "nonces": [ "8d940a5ec4ad122394cd2596ecfbf933a8d8fb0196015cc0a35399e3c326758c" ]
}
```

public_keys An array of hex-encoded compressed public keys for blinding the given scripts. Must be present if "blinding_keys_required" was `true` in the request, and absent otherwise.

nonces An array of hex-encoded 256 bit blinding nonces.

CHAPTER 5

Indices and tables

- `genindex`
- `search`

G

- GA_ack_system_message (*C function*), 13
- GA_auth_handler_call (*C function*), 16
- GA_auth_handler_get_status (*C function*), 14
- GA_auth_handler_request_code (*C function*), 15
- GA_auth_handler_resolve_code (*C function*), 15
- GA_broadcast_transaction (*C function*), 11
- GA_change_settings (*C function*), 14
- GA_change_settings_twofactor (*C function*), 16
- GA_connect (*C function*), 2
- GA_convert_amount (*C function*), 9
- GA_create_session (*C function*), 1
- GA_create_subaccount (*C function*), 5
- GA_create_transaction (*C function*), 10
- GA_destroy_auth_handler (*C function*), 16
- GA_destroy_json (*C function*), 14
- GA_destroy_session (*C function*), 2
- GA_destroy_string (*C function*), 17
- GA_disable_all_pin_logins (*C function*), 10
- GA_encrypt_with_pin (*C function*), 9
- GA_generate_mnemonic (*C function*), 18
- GA_generate_mnemonic_12 (*C function*), 18
- GA_get_assets (*C function*), 3
- GA_get_available_currencies (*C function*), 9
- GA_get_balance (*C function*), 9
- GA_get_credentials (*C function*), 13
- GA_get_fee_estimates (*C function*), 12
- GA_get_networks (*C function*), 19
- GA_get_previous_addresses (*C function*), 7
- GA_get_proxy_settings (*C function*), 2
- GA_get_random_bytes (*C function*), 18
- GA_get_receive_address (*C function*), 7
- GA_get_settings (*C function*), 14
- GA_get_subaccount (*C function*), 6
- GA_get_subaccounts (*C function*), 5
- GA_get_system_message (*C function*), 13
- GA_get_thread_error_details (*C function*), 1
- GA_get_transaction_details (*C function*), 8
- GA_get_transactions (*C function*), 6
- GA_get_twofactor_config (*C function*), 14
- GA_get_uniform_uint32_t (*C function*), 19
- GA_get_unspent_outputs (*C function*), 7
- GA_get_unspent_outputs_for_private_key (*C function*), 8
- GA_get_wallet_identifier (*C function*), 3
- GA_get_watch_only_username (*C function*), 4
- GA_http_request (*C function*), 3
- GA_init (*C function*), 1
- GA_login_user (*C function*), 4
- GA_psbt_get_details (*C function*), 11
- GA_psbt_sign (*C function*), 10
- GA_reconnect_hint (*C function*), 2
- GA_refresh_assets (*C function*), 3
- GA_register_network (*C function*), 18
- GA_register_user (*C function*), 4
- GA_remove_account (*C function*), 5
- GA_rename_subaccount (*C function*), 6
- GA_send_nlocktimes (*C function*), 11
- GA_send_transaction (*C function*), 11
- GA_set_csvtime (*C function*), 12
- GA_set_nlocktime (*C function*), 12
- GA_set_notification_handler (*C function*), 1
- GA_set_transaction_memo (*C function*), 12
- GA_set_unspent_outputs_status (*C function*), 8
- GA_set_watch_only (*C function*), 4
- GA_sign_transaction (*C function*), 10
- GA_twofactor_cancel_reset (*C function*), 17
- GA_twofactor_change_limits (*C function*), 17
- GA_twofactor_reset (*C function*), 16
- GA_twofactor_undo_reset (*C function*), 16
- GA_update_subaccount (*C function*), 6
- GA_validate_asset_domain_name (*C function*), 4
- GA_validate_mnemonic (*C function*), 18