IOTA Entangled

Oct 15, 2019

Content

Entangled is a monorepo containing all you need to get IOTA operating in C/C++ Containing IOTA components, models as well as cryptography primitives used in IOTA.

Applications:

- *IOTA CClient Library*: A low level IOTA client implementation
- MAM: A low level implementation for MAM
- CIRI: A low level implementation of IOTA node largely inspired by IRI
- Tanglescope: A monitoring tool for the Tangle

CHAPTER 1

IOTA CClient Library

An IOTA client library implementation in C language, aka iota.c. The C library is able to run on a small device that supports full TCP/IP stack.

CClient Github

Example projects for IoT application:

- iota_esp32_wallet: IOTA Wallet application.
- iota_esp32_cashier: A portable payment device.
- tangle_pigeon_esp32: TanglePigeon carries a message to the Tangle.

1.1 Getting Start

Before using client APIs, you need to initialize client service. Here shows how to call getNodeInfo in CClient. More details please see: cclient_example

```
retcode_t ret = RC_ERROR;
iota_client_service_t serv;
serv.http.path = "/";
serv.http.content_type = "application/json";
serv.http.accept = "application/json";
// IRI endpoint
serv.http.host = "nodes.thetangle.org";
// IRI API port number
serv.http.port = 443;
serv.http.api_version = 1;
serv.serializer_type = SR_JSON;
// needed for HTTPS
serv.http.ca_pem = amazon_ca1_pem;

// initialize core APIs
iota_client_core_init(&serv);
```

(continues on next page)

(continued from previous page)

```
// initialize extended APIs
iota_client_extended_init();

// new and check the response object
get_node_info_res_t *node_res = get_node_info_res_new();
if (node_res == NULL) {
  printf("Error: OOM\n");
  return;
}

// calls getNodeInfo
ret = iota_client_get_node_info(s, node_res);
printf("%s", error_2_string(ret));

// clean up
get_node_info_res_free(&node_res);
iota_client_extended_destroy();
iota_client_core_destroy(&serv);
```

1.2 Client Service

Unnamed Group

retcode_t **iota_client_service_init** (*iota_client_service_t * serv*)
init CClient service

Return error code

Parameters

- `serv`: service object

void **iota_client_service_destroy** (*iota_client_service_t * serv*)
destory and clean CClient service

Parameters

- `serv`: service object

struct http_info_t
#include <service.h> HTTP request information.

Public Members

char const* **host**
Host name

char const* **path**
The path of HTTP/HTTPS request

char const* **content_type**
Content type of request

char const* **accept**
Accept content type of response

char const* **ca_pem**
String of root ca

uint16_t **port**
Port number of the host

int **api_version**
Number of IOTA API version

struct iota_client_service_t
#include <service.h> client service

Public Members

http_info_t **http**
The http request information

serializer_t **serializer**
The client serializer

serializer_type_t **serializer_type**

1.3 Core APIs

- *iota_client_add_neighbors*
- *iota_client_attach_to_tangle*
- *iota_client_broadcast_transactions*
- *iota_client_check_consistency*
- *iota_client_find_transactions*
- *iota_client_get_balances*
- *iota_client_get_inclusion_states*
- *iota_client_get_neighbors*
- *iota_client_get_node_api_conf*
- *iota_client_get_node_info*
- *iota_client_get_tips*
- *iota_client_get_transactions_to_approve*
- *iota_client_get_trytes*
- *iota_client_remove_neighbors*
- *iota_client_store_transactions*
- *iota_client_were_addresses_spent_from*

retcode_t **iota_client_add_neighbors**(*iota_client_service_t* const *const *service*, *add_neighbors_req_t* const *const *req*, *add_neighbors_res_t* * *res*)
Add a list of neighbors to the IRI node.

Assumes **addNeighbors** command is availiable on the node. It should be noted that this is only temporary, and the added neighbors will be removed from your set of neighbors after you relaunch IRI.

The URI format:

- udp://IPADDRESS:PORT
- tcp://IPADDRESS:PORT

Return retcode_t

Parameters

- `[in] service`: client service
- `[in] req`: A list of URIs.
- `[out] res`: Number of neighbors that were added.

retcode_t **iota_client_attach_to_tangle**(*iota_client_service_t* const *const *service*, *attach_to_tangle_req_t* const *const *req*, *attach_to_tangle_res_t* * *res*)

Performs Proof-of-Work required to attach a transation to the Tangle.

Prepares the specified transactions (trytes) for attachment to the Tangle by doing Proof of Work. You need to supply **branch transaction** as well as **trunk transaction**. These are the tips which you're going to validate and reference with this transaction. These are obtainable by the *iota_client_get_transactions_to_approve* API call.

The returned value is a different set of tryte values which you can input into *iota_client_broadcast_transactions* and *iota_client_store_transactions*

Returns list of transaction trytes and overwrites the following fields:

- `hash`
- `nonce`
- `attachmentTimestamp`
- `attachmentTimestampLowerBound`
- `attachmentTimestampUpperBound`

Return retcode_t

Parameters

- `[in] service`: client service
- `[in] req`: Request containing: trunk, branch, minWeightMagnitude, trytes.
- `[out] res`: Array of transaction trytes with nonce and attachment timestamps

example_attach_to_tangle

retcode_t **iota_client_broadcast_transactions**(*iota_client_service_t* const *const *service*, *broadcast_transactions_req_t* * *req*)

Broadcast a list of transactions to all node's neighbors.

These trytes are returned by *iota_client_attach_to_tangle*, or by doing proof of work somewhere else.

Return retcode_t

Parameters

- `[in] service`: client service
- `[in] req`: The list of transaction trytes to broadcast

example_broadcast_transactions

retcode_t **iota_client_check_consistency**(*iota_client_service_t* const *const service*, *check_consistency_req_t* const *const req*, *check_consistency_res_t* * *res*)

Check the consistency of the subtangle formed by the provided tails.

An inconsistent transaction will not be accepted by the network, and a reattachment is required by *iota_client_replay_bundle*.

Return retcode_t

Parameters

- `[in] service`: client service
- `[in] req`: The tails describing the subtangle.
- `[out] res`: Consistency state of given transaction or co-consistency of given transactions.

example_check_consistency

retcode_t **iota_client_find_transactions**(*iota_client_service_t* const *const service*, *find_transactions_req_t* const *const req*, *find_transactions_res_t* *const *res*)

Searches transactions.

It allows to search transactions by **addresses, tags, and approvees**

Return retcode_t

Parameters

- `[in] service`: client service
- `[in] req`: The query objects
- `[out] res`: Transaction hashes

example_find_transactions

retcode_t **iota_client_get_balances**(*iota_client_service_t* const *const service*, *get_balances_req_t* const *const req*, *get_balances_res_t* * *res*)

Fetches **confirmed**

If the **tips** was not provided, the returned balance is correct as of the latest confirmed milestone.

Return retcode_t

Parameters

- `[in] service`: client service
- `[in] req`: Address, tips and a threshold for balance calculation.
- `[out] res`: Returned balances list and the referencing tips (or milestone).

example_get_balance

retcode_t **iota_client_get_inclusion_states**(*iota_client_service_t* const *const service*, *get_inclusion_states_req_t* const *const req*, *get_inclusion_states_res_t* * *res*)

Fetches inclusion states of given list of transactions.

Get the inclusion states of a set of transactions. This is for determining if a transaction was accepted and confirmed by the network or not. You can search for multiple tips (and thus, milestones) to get past inclusion states of transactions.

This API call returns a list of boolean values in the same order as the submitted transactions. Boolean values will be **true** for confirmed transactions, otherwise **false**.

Return retcode_t

Parameters

- `[in] service`: client service
- `[in] req`: Transactions and Tips that you want to search for the inclusion state.
- `[out] res`: A list of inclusion state.

example_get_inclusion_states

retcode_t **iota_client_get_neighbors**(*iota_client_service_t* const *const service*, *get_neighbors_res_t* *const *res*)

Returns a list of connected neighbors.

Return retcode_t

Parameters

- `[in] service`: client service
- `[out] res`: A list of neighbors.

retcode_t **iota_client_get_node_api_conf**(*iota_client_service_t* const *const service*, *get_node_api_conf_res_t* * *res*)

Returns configuration of the connected node.

Return retcode_t

Parameters

- `[in] service`: client service
- `[out] res`: node configuration

example_node_api_conf

retcode_t **iota_client_get_node_info**(*iota_client_service_t* const *const service*, *get_node_info_res_t* * *res*)

Returns information about connected node.

Return retcode_t

Parameters

- `[in] service`: client service
- `[out] res`: node information

example_node_info

retcode_t **iota_client_get_tips**(*iota_client_service_t* const *const *service*, *get_tips_res_t* *const *res*)
Returns a list of tips (transactions not referenced by other transactions).

Return retcode_t

Parameters

- `[in] service`: client service
- `[out] res`: A list of tip transactions.

example_get_tips

retcode_t **iota_client_get_transactions_to_approve**(*iota_client_service_t* const *const *service*, *get_transactions_to_approve_req_t* const *const *req*, *get_transactions_to_approve_res_t* * *res*)
Does the **tip selection**

Tip selection which returns **trunk** and **branch** transaction. The input value **depth** determines how many milestones to go back for finding the transactions to approve. The higher your **depth** value, the more work you have to do as you are confirming more transactions. If the **depth** is too large (usually above 15, it depends on the node's configuration) an error will be returned. The **reference** is an optional hash of a transaction you want to approve. If it can't be found at the specified **depth** then an error will be returned.

Return retcode_t

Parameters

- `[in] service`: client service
- `[in] req`: the **depth** and a **reference(optional)**
- `[out] res`: A pair of approved transactions(trunk and branch transaction)

example_get_transactions_to_approve

retcode_t **iota_client_get_trytes**(*iota_client_service_t* const *const *service*, *get_trytes_req_t* const *const *req*, *get_trytes_res_t* * *res*)
Fetches transaction trytes by a given list of transaction hashes.

The transaction trytes can be converted into transaction object(#iota_transaction_t) by calling #transaction_deserialize_from_trits or #transaction_deserialize.

Return retcode_t

Parameters

- `[in] service`: client service
- `[in] req`: Transaction hashes
- `[out] res`: Transaction trytes

example_get_trytes

retcode_t **iota_client_remove_neighbors**(*iota_client_service_t* const *const *service*, *remove_neighbors_req_t* const *const *req*, *remove_neighbors_res_t* * *res*)
Removes a list of neighbors from the connected IRI node.

Assumes removeNeighbors command is available on the node. This method has temporary effect until your IRI node relaunches.

The URI format:

- udp://IPADDRESS:PORT
- tcp://IPADDRESS:PORT

Return retcode_t

Parameters

- `[in] service`: client service
- `[in] req`: List of URIs that you want to remove from the node.
- `[out] res`: Number of neighbors that were removed.

retcode_t **iota_client_store_transactions**(*iota_client_service_t* const *const *service*, *store_transactions_req_t* const *const *req*)

Store transactions in an IRI node's local storage.

The trytes must be validated by *iota_client_attach_to_tangle*

Return retcode_t

Parameters

- `[in] service`: client service
- `[in] req`: List of transaction trytes

example_store_transactions

retcode_t **iota_client_were_addresses_spent_from**(*iota_client_service_t* const *const *service*, *were_addresses_spent_from_req_t* const *const *req*, *were_addresses_spent_from_res_t* * *res*)

Check if a list of addresses was ever spent from, in the current epoch, or in previous epochs.

Return retcode_t

Parameters

- `[in] service`: client service
- `[in] req`: A list of address hash that you want to query for the status.
- `[inout] res`: A list of address state.

example_were_addresses_spent_from

1.4 Extended APIs

- *iota_client_broadcast_bundle*
- *iota_client_find_transaction_objects*
- *iota_client_get_account_data*
- *iota_client_get_bundle*

- *iota_client_get_inputs*
- *iota_client_get_latest_inclusion*
- *iota_client_get_new_address*
- *iota_client_get_transaction_objects*
- *iota_client_is_promotable*
- *iota_client_prepare_transfers*
- *iota_client_promote_transaction*
- *iota_client_replay_bundle*
- *iota_client_send_transfer*
- *iota_client_send_trytes*
- *iota_client_store_and_broadcast*
- *iota_client_traverse_bundle*

retcode_t **iota_client_broadcast_bundle**(*iota_client_service_t* const *const *serv*, *flex_trit_t* const *const *tail_hash*, *bundle_transactions_t* *const *bundle*)

Re-broadcasts all transactions in a bundle.

Re-broadcasts all transactions in a bundle given the tail transaction hash. It might be useful when transactions did not properly propagate, particularly in the case of large bundles.

Return retcode_t

Parameters

- `[in] serv`: client service
- `[in] tail_hash`: A tail transaction hash
- `[out] bundle`: A transaction bundle

example_broadcast_bundle

retcode_t **iota_client_find_transaction_objects**(*iota_client_service_t* const *const *serv*, *find_transactions_req_t* const *const *tx_queries*, transaction_array_t * *out_tx_objs*)

Query transaction object by addresses, tags, and approvees.

Wrapper function for *iota_client_find_transactions* and *iota_client_get_trytes*. Searches for transactions given a query object with **addresses**, **tags** and **approvees** fields. Multiple query fields are supported and it returns intersection of results.

Return retcode_t

Parameters

- `[in] serv`: client service
- `[in] tx_queries`: transactions to query
- `[out] out_tx_objs`: A list of transation objects

example_find_transaction_objects

retcode_t **iota_client_get_account_data**(*iota_client_service_t* const *const serv, *flex_trit_t* const *const seed, uint8_t security, *account_data_t* * out_account)

Returns an *account_data_t* object.

Returns an *account_data_t* object, containing account information about **addresses**, **transactions**, **inputs** and total account balance.

Return retcode_t

Parameters

- `[in] serv`: client service
- `[in] seed`: seed
- `[in] security`: Security level to be used for getting addresses
- `[out] out_account`: the result containing a unused address, used addresses, transactions, and total balance.

example_get_account_data

struct account_data_t

Account information.

Public Members

uint64_t **balance**

total balance

flex_trit_t account_data_t::latest_address[FLEX_TRIT_SIZE_243]

unused address

hash243_queue_t **addresses**

List of used addresses

hash243_queue_t **transactions**

List of transactions

UT_array* **balances**

List of balances

retcode_t **iota_client_get_bundle**(*iota_client_service_t* const *const serv, *flex_trit_t* const *const tail_hash, *bundle_transactions_t* *const bundle, *bundle_status_t* *const bundle_status)

Gets the associated bundle transactions from a tail transaction.

Fetches and validates the bundle given a **tail** transaction hash, by calling *iota_client_traverse_bundle*

Return retcode_t

Parameters

- `[in] serv`: client service
- `[in] tail_hash`: Tail transaction hash
- `[out] bundle`: A Bundle containing transaction objects
- `[out] bundle_status`: Status of bundle validation

example_get_bundle

retcode_t **iota_client_get_inputs**(*iota_client_service_t* const *const *serv*, *flex_trit_t* const *const *seed*, address_opt_t const *addr_opt*, int64_t const *threshold*, *inputs_t* *const *out_inputs*)

Gets inputs by generating address and fetching balances.

Create an *inputs_t* object contaning a list of *input_t* and total balance.

Return retcode_t

Parameters

- `[in] serv`: client service
- `[in] seed`: A seed for address generator
- `[in] addr_opt`: address information
- `[in] threshold`: Minimum balance required
- `[out] out_inputs`: Inputs object containing a list of *input_t* and total balance

example_get_inputs

retcode_t **iota_client_get_latest_inclusion**(*iota_client_service_t* const *const *serv*, hash243_queue_t const *transactions*, *get_inclusion_states_res_t* * *out_states*)

Gets inclusion states.

Fetches inclusion states of given transactions and a list of tips, by calling *iota_client_get_inclusion_states* and using **latestSolidSubtangleMilestone** as the tip.

Return retcode_t

Parameters

- `[in] serv`: client service
- `[in] transactions`: List of transaction hashes
- `[out] out_states`: List of inclusion states

example_get_latest_inclusion

retcode_t **iota_client_get_new_address**(*iota_client_service_t* const *const *serv*, *flex_trit_t* const *const *seed*, address_opt_t const *addr_opt*, hash243_queue_t * *out_addresses*)

Generates and returns addresses including an unused address.

Generates and returns addresses by calling *iota_client_find_transactions* until the first unused address is detected.

Return retcode_t

Parameters

- `[in] serv`: client service
- `[in] seed`: A seed for address generation.
- `[in] addr_opt`: address information containing security level and key indices
- `[out] out_addresses`: List of addresses including an unused address.

example_get_new_address

retcode_t **iota_client_get_transaction_objects**(*iota_client_service_t* const *const serv, get_trytes_req_t *const tx_hashes, transaction_array_t * out_tx_objs*)

Fetches the transaction objects, given a list of transaction hashes.

Return retcode_t

Parameters

- `[in] serv`: client service
- `[in] tx_hashes`: List of transaction hashes
- `[out] out_tx_objs`: List of transaction objects

retcode_t **iota_client_is_promotable**(*iota_client_service_t* const *const serv, flex_trit_t* const *const tail_tx*, bool * *out_promotable*)

Checks if a tail transaction is **promotable**

Checks if a transaction is **promotable**, by calling *iota_client_check_consistency* and verifying that **attachment_timestamp** is above a lower bound in the transaction. Lower bound is calculated based on number of milestones issued since transaction attachment.

Return retcode_t

Parameters

- `[in] serv`: client service
- `[in] tail_tx`: Tail transaction hash
- `[out] out_promotable`: **True** if it is, otherwise **False**

example_is_promotable

retcode_t **iota_client_prepare_transfers**(*iota_client_service_t* const *const serv, flex_trit_t* const *const seed*, uint8_t *security*, transfer_array_t const *const *transfers*, *flex_trit_t* const *const remainder*, *inputs_t* const *const inputs*, bool *validate_inputs*, uint64_t *timestamp*, *bundle_transactions_t* * *out_bundle*)

Prepares the transaction trytes by generating a bundle.

Return retcode_t

Parameters

- `[in] serv`: client service
- `[in] seed`: A seed
- `[in] security`: The security level of addresses, value could be 1,2,3.
- `[in] transfers`: A list of transfer objects.
- `[in] remainder`: Optional, a remainder address.
- `[in] inputs`: Optional, A list of input objects.
- `[in] validate_inputs`: True for validating input balances, otherwise will not validate input balances.
- `[in] timestamp`: Optional, a timestamp in ms.

- `[inout] out_bundle`: A list of bundle transaction objects.

example_prepare_transfer

retcode_t **iota_client_promote_transaction**(*iota_client_service_t* const *const serv, *flex_trit_t* const *const *tail_hash*, uint8_t *security*, uint32_t const *depth*, uint8_t const *mwm*, transfer_array_t * *spam_transfers*, *bundle_transactions_t* * *out_bundle*)

Promotes a transaction by adding zero-value spam transactions on top of it.

Attempts to promote a transaction using a provided transfers and, if successful. This will effectively double the chances of the transaction to be picked, and this approved.

Return retcode_t

Parameters

- `[in] serv`: client service
- `[in] tail_hash`: A tail hash which is the transaction in a bundle with **current_index = 0**.
- `[in] security`: security level of the address.
- `[in] depth`: The depth as which Random Walk starts, **3** is a typicall value used by wallets.
- `[in] mwm`: Mininum number of trailing zeros in transaction hash, **14** for mainnet and **9** for testnet.
- `[in] spam_transfers`: A list of transfers to promote with.
- `[out] out_bundle`: List of transactions made with the attached transaction objects.

example_promote_transacion

retcode_t **iota_client_replay_bundle**(*iota_client_service_t* const *const *serv*, *flex_trit_t* const *const *tail_hash*, uint32_t const *depth*, uint8_t const *mwm*, *flex_trit_t* const *const *reference*, *bundle_transactions_t* *const *bundle*)

Reattaches a transfer to the Tangle by selecting tips and performing the Proof-of-Work again.

Reattachments are useful in case the original transactions are pending, and can be done securely as many times as needed. This will make a new, but identical transaction which now also can be approved. If any of the replayed transactions gets approved, the others stop getting approved.

Return retcode_t

Parameters

- `[in] serv`: client service
- `[in] tail_hash`: Tail transaction hash
- `[in] depth`: The depth for getting transactions to approve
- `[in] mwm`: The Minimum Weight Magnitude for doing Proof-of-Work
- `[in] reference`: Hash of transaction to start random-walk from. Can be **Null**, in that case the latest milestone is used as a reference.
- `[out] bundle`: Analyzed transaction objects.

example_replay_bundle

retcode_t **iota_client_send_transfer**(*iota_client_service_t* const *const serv, *flex_trit_t* const *const seed, uint8_t security, uint32_t const depth, uint8_t const mwm, bool local_pow, transfer_array_t const *const transfers, *flex_trit_t* const *const remainder_address, *flex_trit_t* const *const reference, *inputs_t* const *const inputs, *bundle_transactions_t* * out_tx_objs)

Wrapper function: Runs *iota_client_prepare_transfers* and *iota_client_send_trytes*.

Return retcode_t

Parameters

- `[in] serv`: Client service.
- `[in] seed`: A seed.
- `[in] security`: The security level of addresses, value could be 1,2,3.
- `[in] depth`: The depth for getting transactions to approve
- `[in] mwm`: The minimum weight magnitude for doing Proof-of-Work
- `[in] local_pow`: If **true** do local Proof-of-Work, otherwise do remote.
- `[in] transfers`: A list of transfer objects.
- `[in] remainder_address`: Optional, a remainder address.
- `[in] reference`: Optional, hash of transaction to start Random-Walk from
- `[in] inputs`: inputs Optional, A list of input objects.
- `[inout] out_tx_objs`: A list of bundle transaction objects.

retcode_t **iota_client_send_trytes**(*iota_client_service_t* const *const *serv*, hash8019_array_p const *trytes*, uint32_t const *depth*, uint8_t const *mwm*, *flex_trit_t* const *const *reference*, bool const *local_pow*, transaction_array_t * *out_transactions*)

Wrapper function: Runs *iota_client_attach_to_tangle* and *iota_client_store_and_broadcast*.

Return retcode_t

Parameters

- `[in] serv`: client service
- `[in] trytes`: List of trytes to attach, store and broadcast
- `[in] depth`: The depth at which Random Walk starts. **3** is typically used by wallets, meaning that RW starts 3 milestones back.
- `[in] mwm`: Minimum number of trailing zeros in transaction hash. This is used to search for a valid **nonce**. Currently it is **14** on mainnet & spamnet and **9** on most other testnets.
- `[in] reference`: Optional, hash of transaction to start Random-Walk from.
- `[in] local_pow`: If **true** do local Proof-of-Work, otherwise do remote.
- `[out] out_transactions`: the transaction objects were sent to IRI node.

example_send_trytes

retcode_t **iota_client_store_and_broadcast**(*iota_client_service_t* const *const *serv*, *store_transactions_req_t* const *const *trytes*)

Stores and broadcasts a list of **attached transaction trytes**.

Note: Persist the transaction trytes in local storage *before* calling this command, to ensure that reattachment is possible, until your bundle has been included. Any transactions stored with this command will eventually be erased, as a result of a snapshot.

Return retcode_t

Parameters

- `[in] serv`: client service
- `[out] trytes`: List of transaction trytes.

retcode_t **iota_client_traverse_bundle**(*iota_client_service_t* const *const *serv*, *flex_trit_t* const *const *tail_hash*, *bundle_transactions_t* *const *bundle*)
Wrapper function: Runs traverse_bundle.

Return retcode_t

Parameters

- `[in] serv`: Client service
- `[in] tail_hash`: Tail transaction hash.
- `[out] bundle`: Bundle as list of transaction objects.

example_traverse_bundle

1.5 Requests

Unnamed Group

typedef struct *add_neighbors_req_s* **add_neighbors_req_t**
The data structure of the add neighbors request.

The URI format:

- tcp://IPADDRESS:PORT
- udp://IPADDRESS:PORT

*add_neighbors_req_t** **add_neighbors_req_new**()
Allocates the add neighbors request object.

Return A pointer to the request object.

void **add_neighbors_req_free**(*add_neighbors_req_t* ** *req*)
Free the add neighbors request.

Parameters

- `[in] req`: The request object.

retcode_t **add_neighbors_req_uris_add**(*add_neighbors_req_t* * *req*, char const *const *uri*)
Add an URI string to the request.

Return retcode_t

Parameters

- `[in] req`: The request object.
- `[in] uri`: An URI string. see *add_neighbors_req_t::uris*

const char* **add_neighbors_req_uris_at** (*add_neighbors_req_t* * *req*, size_t *idx*)
Get an URI string from the URI list.

Return A string of URI. see *add_neighbors_req_t::uris*

Parameters

- `[in] req`: The request object
- `[in] idx`: The index of URI list.

size_t **add_neighbors_req_uris_len** (*add_neighbors_req_t* * *req*)
The size of URI list.

Return The number of URI elements in the list.

Parameters

- `[in] req`: request object

struct add_neighbors_req_s
#include <add_neighbors.h> The data structure of the add neighbors request.

The URI format:

- tcp://IPADDRESS:PORT
- udp://IPADDRESS:PORT

Public Members

UT_array* **uris**
List of neighbor URIs for adding

Unnamed Group

ATTACH_TO_TANGLE_MAIN_MWM

ATTACH_TO_TANGLE_TEST_MWM

*attach_to_tangle_req_t** **attach_to_tangle_req_new** ()
Allocates an attach to tangle request.

Return A pointer to the request object.

void **attach_to_tangle_req_free** (*attach_to_tangle_req_t* ** *req*)
Free a request object.

Parameters

- `[in] req`: The request object to be freed.

void **attach_to_tangle_req_init** (*attach_to_tangle_req_t* * *req*, *flex_trit_t* const *const *trunk*, *flex_trit_t* const *const *branch*, uint8_t *mwm*)
Set trunk, branch, and mwm to the request.

Parameters

- `[in] req`: The request object.
- `[in] trunk`: Trunk transaction hash.
- `[in] branch`: Branch transaction hash.
- `[in] mwm`: Minimum Weight Magnitude for Proof-of-Work.

retcode_t **attach_to_tangle_req_trytes_add**(*attach_to_tangle_req_t* * *req*, *flex_trit_t* const *const *raw_trytes*)
Set transaction trytes to the request object.

Return retcode_t

Parameters

- `[in] req`: The request object.
- `[in] raw_trytes`: A transaction trytes.

*flex_trit_t** **attach_to_tangle_req_trytes_get**(*attach_to_tangle_req_t* * *req*, size_t *index*)
Get transaction trytes by index.

Return Null on failed, otherwise return a pointer of flex_trit_t.

Parameters

- `[in] req`: The attach to tangle request object.
- `[in] index`: The index of transaction trytes list.

static *flex_trit_t** **attach_to_tangle_req_trunk**(*attach_to_tangle_req_t* *const *req*)
Get trunk hash from the request object.

Return The pointer of trunk transaction hash.

Parameters

- `[in] req`: The attach to tangle request object.

static *flex_trit_t** **attach_to_tangle_req_branch**(*attach_to_tangle_req_t* *const *req*)
Get branch hash from the request object.

Return The pointer of branch transaction hash.

Parameters

- `[in] req`: The attach to tangle request object.

static uint8_t **attach_to_tangle_req_mwm**(*attach_to_tangle_req_t* *const *req*)
Get MWM from the request object.

Return The value of MWM.

Parameters

- `[in] req`: The attach to tangle request object.

struct attach_to_tangle_req_t
#include <attach_to_tangle.h> The data structure of the attach to tangle request.

Public Members

hash8019_array_p **trytes**
List of trytes (raw transaction data) to attach to the tangle.

uint8_t **mwm**
Min Weight Magnitude,Proof of Work intensity. The value for mainnet is 14 and for others is 9.

flex_trit_t attach_to_tangle_req_t::trunk[FLEX_TRIT_SIZE_243]
Trunk transaction to approve

flex_trit_t attach_to_tangle_req_t::branch[FLEX_TRIT_SIZE_243]
branch transaction to approve

Unnamed Group

typedef struct *broadcast_transactions_req_s* **broadcast_transactions_req_t**
The data structure of the broadcast transactions request.

*broadcast_transactions_req_t** **broadcast_transactions_req_new**()
Allocates broadcast transactions request.

Return A pointer to broadcast transactions request object.

void **broadcast_transactions_req_free**(*broadcast_transactions_req_t* **const *req*)
Free the request object.

Parameters

- `[in] req`: The request object

retcode_t **broadcast_transactions_req_trytes_add**(*broadcast_transactions_req_t* * *req*, *flex_trit_t* const *const *raw_trytes*)
Add a transaction trytes to the request object.

Return retcode_t

Parameters

- `[in] req`: The request object.
- `[in] raw_trytes`: A transaction trytes.

*flex_trit_t** **broadcat_transactions_req_trytes_get**(*broadcast_transactions_req_t* * *req*, size_t *index*)
Get a transaction trytes by index.

Return A pointer to transaction trytes.

Parameters

- `[in] req`: The request object
- `[in] index`: The index of list.

struct broadcast_transactions_req_s
#include <broadcast_transactions.h> The data structure of the broadcast transactions request.

Public Members

hash8019_array_p **trytes**
List of valid transaction trytes to be broadcasted

Unnamed Group

typedef struct *check_consistency_req_s* **check_consistency_req_t**
The data structure of the check consistency request.

*check_consistency_req_t** **check_consistency_req_new**()
Allocates a check consistency request object.

Return A pointer of the request object.

void **check_consistency_req_free**(*check_consistency_req_t* ** *req*)
Free the request object.

Parameters

- `[in] req`: the request object.

static *retcode_t* **check_consistency_req_tails_add**(*check_consistency_req_t* *const *req*, *flex_trit_t* const *const *hash*)
Add a tail transaction to the request.

Return retcode_t

Parameters

- `[in] req`: The request object.
- `[in] hash`: The hash of a tail transaction.

static *flex_trit_t** **check_consistency_req_tails_get**(*check_consistency_req_t* *const *req*, size_t *index*)
Get the size of the tail transactions.

Return A pointer to a tail transaction hash.

Parameters

- `[in] req`: The request object.
- `[in] index`: The index of the tail transaction list.

struct check_consistency_req_s
#include <*check_consistency.h*> The data structure of the check consistency request.

Public Members

hash243_queue_t **tails**
Tail transaction hash (hash of transaction with **currentIndex=0**), or list of tail transaction hashes.

Unnamed Group

typedef struct *find_transactions_req_s* **find_transactions_req_t**
The data structure of the find transactions request.

*find_transactions_req_t** **find_transactions_req_new**()
Allocates a find transactions request object.

Return A pointer to a request object.

void **find_transactions_req_free**(*find_transactions_req_t* ** *req*)
Free a find transactions request object.

Parameters

- `[in] req`: The request object.

static *retcode_t* **find_transactions_req_bundle_add**(*find_transactions_req_t* *const *req*, *flex_trit_t* const *const *hash*)
Add a bundle hash to request object.

Return retcode_t

Parameters

- `[in] req`: The request object.
- `[in] hash`: A bundle hash.

static *flex_trit_t** **find_transactions_req_bundle_get**(*find_transactions_req_t* *const *req*, size_t *index*)
Get a bundle hash by index.

Return A pointer to a bundle hash.

Parameters

- `[in] req`: The request object.
- `[in] index`: The index of the bundle list.

static *retcode_t* **find_transactions_req_address_add**(*find_transactions_req_t* *const *req*, *flex_trit_t* const *const *hash*)
Add an address hash to request object.

Return retcode_t

Parameters

- `[in] req`: The request object.
- `[in] hash`: An address hash.

static *flex_trit_t** **find_transactions_req_address_get**(*find_transactions_req_t* *const *req*, size_t *index*)
Get an address by index.

Return A pointer to an address hash.

Parameters

- `[in] req`: The request object
- `[in] index`: The index of the address list.

static *retcode_t* **find_transactions_req_tag_add** (*find_transactions_req_t* *const *req*, *flex_trit_t* const *const *hash*)

Add a tag to request object.

Return retcode_t

Parameters

- `[in] req`: The request object.
- `[in] hash`: A tag hash.

static *flex_trit_t** **find_transactions_req_tag_get** (*find_transactions_req_t* *const *req*, size_t *index*)

Get a tag by index.

Return A pointer to a tag.

Parameters

- `[in] req`: The request object.
- `[in] index`: The index of the tag list.

static *retcode_t* **find_transactions_req_approvee_add** (*find_transactions_req_t* *const *req*, *flex_trit_t* const *const *hash*)

Add an approvee to request object.

Return retcode_t

Parameters

- `[in] req`: The request object.
- `[in] hash`: An approvee hash.

static *flex_trit_t** **find_transactions_req_approvee_get** (*find_transactions_req_t* *const *req*, size_t *index*)

Get an approvee by index.

Return A pointer to an approvee hash.

Parameters

- `[in] req`: The request object.
- `[in] index`: The index of the approvee list.

struct find_transactions_req_s

#include <find_transactions.h> The data structure of the find transactions request.

Public Members

hash243_queue_t **bundles**

List of bundle hashes. Transactions belonging to bundles will be returned

hash243_queue_t **addresses**
List of addresses. Transactions with any of these addresses as input/output will be returned

hash81_queue_t **tags**
List of transaction tags. Transactions with any of these tags will be returned

hash243_queue_t **approvees**
List of approvees of a transaction. Transactions which directly approve any of approvees will be returned

Unnamed Group

*get_balances_req_t** **get_balances_req_new**()
Allocates a get balances request object.

Return A pointer to the request.

void **get_balances_req_free**(*get_balances_req_t* ** *req*)
Free a get balances request object.

Parameters

- `[in] req`: The request object.

static *retcode_t* **get_balances_req_address_add**(*get_balances_req_t* *const *req*, *flex_trit_t* const *const *hash*)
Add an address hash to a get balances request.

Return retcode_t

Parameters

- `[in] req`: The request object.
- `[in] hash`: An address hash.

static *flex_trit_t** **get_balances_req_address_get**(*get_balances_req_t* *const *req*, size_t *index*)
Get an address by index.

Return A pointer to an address hash.

Parameters

- `[in] req`: The request object.
- `[in] index`: The index of the address list.

static *retcode_t* **get_balances_req_tip_add**(*get_balances_req_t* *const *req*, *flex_trit_t* const *const *hash*)
Add a tip transaction to the request object.

Return retcode_t

Parameters

- `[in] req`: The request object.
- `[in] hash`: A tips transaction hash.

static *flex_trit_t** **get_balances_req_tip_get**(*get_balances_req_t* *const *req*, size_t *index*)
Get a tip transaction by index.

Return A pointer to a tip transaction hash.

Parameters

- `[in] req`: The request object.
- `[in] index`: The index of the tip transaction list.

struct get_balances_req_t
#include <get_balances.h> The data structure of the get balances request.

Public Members

uint8_t **threshold**
The confirmation threshold between 0 and 100(inclusive). Should be set to 100 for getting balance by counting only confirmed transactions.

hash243_queue_t **addresses**
List of addresses you want to get the confirmed balance for.

hash243_queue_t **tips**
List of hashes, if present calculate the balance of addresses from the PoV of these transactions, can be used to chain bundles.

Unnamed Group

*get_inclusion_states_req_t** **get_inclusion_states_req_new**()
Allocates a get inclusion states request object.

Return A pointer to the request.

void **get_inclusion_states_req_free**(*get_inclusion_states_req_t* ** *req*)
Free a get inclusion states request object.

Parameters

- `[in] req`: The request object.

static *retcode_t* **get_inclusion_states_req_hash_add**(*get_inclusion_states_req_t* *const *req*, *flex_trit_t* const *const *hash*)
Add a transaction hash to the request object.

Return retcode_t

Parameters

- `[in] req`: The request object.
- `[in] hash`: A transaction hash.

static *flex_trit_t** **get_inclusion_states_req_hash_get**(*get_inclusion_states_req_t* *const *req*, size_t *index*)
Get a transation hash by index.

Return A pointer to a transaction hash.

Parameters

- `[in] req`: The request object.
- `[in] index`: An index of the transaction hash list.

static *retcode_t* **get_inclusion_states_req_tip_add**(*get_inclusion_states_req_t* *const *req*, *flex_trit_t* const *const *hash*)

Add a tip transaction to the request object.

Return retcode_t

Parameters

- `[in] req`: The request object.
- `[in] hash`: A tip transaction hash.

static *flex_trit_t** **get_inclusion_states_req_tip_get**(*get_inclusion_states_req_t* *const *req*, size_t *index*)

Get a tip hash by index.

Return A pointer to a tip transaction hash.

Parameters

- `[in] req`: The request object.
- `[in] index`: An index of the tip transaction hash list.

struct get_inclusion_states_req_t

#include <get_inclusion_states.h> The data structure of the get inclusion states request.

Public Members

hash243_queue_t **transactions**

List of transactions you want to get the inclusion state for.

hash243_queue_t **tips**

List of tips (including milestones) you want to search for the inclusion state.

Unnamed Group

typedef struct *get_transactions_to_approve_req_s* **get_transactions_to_approve_req_t**

The data structure of the get transaction to approve.

*get_transactions_to_approve_req_t** **get_transactions_to_approve_req_new**()

Allocates a get transactions to approve request object.

Return A pointer to the request.

void **get_transactions_to_approve_req_free**(*get_transactions_to_approve_req_t* **const *req*)

Free a get transactions to approve request object.

Parameters

- `[in] req`: The request object.

void **get_transactions_to_approve_req_set_depth**(*get_transactions_to_approve_req_t* *const *req*, uint32_t const *depth*)

Set the depth of the request.

Parameters

- `[in] req`: The request object.
- `[in] depth`: The depth value for Random Walk start from.

retcode_t **get_transactions_to_approve_req_set_reference**(*get_transactions_to_approve_req_t* *const *req*, *flex_trit_t* const *const *reference*)

Set a reference transaction hash for the Random Walk start from.

Return RC_OK or RC_OOM.

Parameters

- `[in] req`: The request object.
- `[in] reference`: A reference transaction hash.

struct get_transactions_to_approve_req_s

#include <get_transactions_to_approve.h> The data structure of the get transaction to approve.

Public Members

uint32_t **depth**

Number of bundles to go back to determine the transactions for approval.

*flex_trit_t** **reference**

Transaction hash from which to start the weighted random walk. Use this parameter to make sure the returned tip transaction hashes approve a given reference transaction.

Unnamed Group

typedef struct *get_trytes_req_s* **get_trytes_req_t**

The data structure of the get trytes request.

*get_trytes_req_t** **get_trytes_req_new**()

Allocates a get trytes request object.

Return A pointer to the request object.

void **get_trytes_req_free**(*get_trytes_req_t* **const *req*)

Free a get trytes request object.

Parameters

- `[in] req`: The request object.

static *retcode_t* **get_trytes_req_hash_add**(*get_trytes_req_t* *const *req*, *flex_trit_t* const *const *hash*)

Add a transaction hash to the request.

Return retcode_t

Parameters

- `[in] req`: The request object.
- `[in] hash`: A transaction hash.

static *flex_trit_t** **get_trytes_req_hash_get** (*get_trytes_req_t* *const *req*, size_t *index*)
Get a transaction hash by index.

Return A pointer to a transaction hash.

Parameters

- `[in] req`: The request object.
- `[in] index`: An index of transaction list.

struct get_trytes_req_s
#include <get_trytes.h> The data structure of the get trytes request.

Public Members

hash243_queue_t **hashes**

Unnamed Group

typedef struct *remove_neighbors_req_s* **remove_neighbors_req_t**
The data structure of the remove neighbors request.

The URI format:

- tcp://IPADDRESS:PORT
- udp://IPADDRESS:PORT

*remove_neighbors_req_t** **remove_neighbors_req_new**()
Allocates a remove neighbors request object.

Return A pointer to the request object.

retcode_t **remove_neighbors_req_add** (*remove_neighbors_req_t* * *req*, char const * *uri*)
Add a string URI to the request.

Return retcode_t

Parameters

- `[in] req`: The request object.
- `[in] uri`: A string of URI. see *remove_neighbors_req_t::uris*

void **remove_neighbors_req_free** (*remove_neighbors_req_t* ** *req*)
Free a remove neighbors request object.

Parameters

- `[in] req`: The request object.

char const* **remove_neighbors_req_uris_at**(*remove_neighbors_req_t* * *req*, size_t *idx*)
Get a URI string by index from the request.

Return A URI string. see *remove_neighbors_req_t::uris*

Parameters

- `[in] req`: The request object.
- `[in] idx`: The index of URI list.

size_t **remove_neighbors_req_uris_len**(*remove_neighbors_req_t* * *req*)
Get the size of URI list.

Return The number of URI elements in the list.

Parameters

- `[in] req`: The request object.

struct remove_neighbors_req_s
#include <*remove_neighbors.h*> The data structure of the remove neighbors request.

The URI format:

- tcp://IPADDRESS:PORT
- udp://IPADDRESS:PORT

Public Members

UT_array* **uris**
List of neighbor URIs for removing

Unnamed Group

typedef struct *store_transactions_req_s* **store_transactions_req_t**
The data structure of the store transactions request.

*store_transactions_req_t** **store_transactions_req_new**()
Allocates a store transactions request.

Return A pointer to the request object.

void **store_transactions_req_free**(*store_transactions_req_t* **const *req*)
Free a store transactions request object.

Parameters

- `[in] req`: The request object.

retcode_t **store_transactions_req_trytes_add**(*store_transactions_req_t* * *req*, *flex_trit_t* const *const *raw_trytes*)
Add a transaction trytes to the request.

Return retcode_t

Parameters

- `[in] req`: The request object.
- `[in] raw_trytes`: A transaction trytes.

*flex_trit_t** **store_transactions_req_trytes_get**(*store_transactions_req_t* * *req*, size_t *index*)
Get a transaction trytes by index.

Return A pointer to the transaction trytes.

Parameters

- `[in] req`: The request object.
- `[in] index`: The index of transaction list.

struct store_transactions_req_s
#include <store_transactions.h> The data structure of the store transactions request.

Public Members

hash8019_array_p **trytes**
List of transaction tytes to be stored

Unnamed Group

typedef struct *were_addresses_spent_from_req_s* **were_addresses_spent_from_req_t**
The data structure of the were addresses spent from request.

*were_addresses_spent_from_req_t** **were_addresses_spent_from_req_new**()
Allocates a were addresses spent from request object.

Return A pointer to the request object.

void **were_addresses_spent_from_req_free**(*were_addresses_spent_from_req_t* **const *req*)
Free a were addresses spent from request object.

Parameters

- `[in] req`: The request object.

static *retcode_t* **were_addresses_spent_from_req_add**(*were_addresses_spent_from_req_t* *const *req*, *flex_trit_t* const *const *address*)
Adding an address hash to the request.

Return retcode_t

Parameters

- `[in] req`: The request object.
- `[in] address`: An address hash.

static *flex_trit_t** **were_addresses_spent_from_req_get**(*were_addresses_spent_from_req_t* *const *req*, size_t *index*)
Get an address hash by index.

Return A pointer to an address hash.

Parameters

- `[in] req`: The request object.
- `[in] index`: An index of addresses list.

struct were_addresses_spent_from_req_s
#include <were_addresses_spent_from.h> The data structure of the were addresses spent from request.

Public Members

hash243_queue_t **addresses**

1.6 Responses

Unnamed Group

typedef struct *add_neighbors_res_s* **add_neighbors_res_t**
The data structure of add neighbors response.

*add_neighbors_res_t** **add_neighbors_res_new**()
Allocates an add neighbors response object.

Return A pointer to the response object.

void **add_neighbors_res_free**(*add_neighbors_res_t* ** *res*)
Frees an add neighbors response object.

Parameters

- `[in] res`: a response object.

struct add_neighbors_res_s
#include <add_neighbors.h> The data structure of add neighbors response.

Public Members

int **added_neighbors**
Numbers of neighbors on this connected node

Unnamed Group

*attach_to_tangle_res_t** **attach_to_tangle_res_new**()
Allocates an attach to tangle response object.

Return A pointer to the response object.

retcode_t **attach_to_tangle_res_trytes_add**(*attach_to_tangle_res_t* * *res*, *flex_trit_t* const *const *trytes*)
Adds a transaction trytes in a response object.

Return retcode_t

Parameters

- `[in] res`: The response object.
- `[in] trytes`: A transaction trytes.

*flex_trit_t** **attach_to_tangle_res_trytes_at** (*attach_to_tangle_res_t * res*, int *index*)
Gets a transaction trytes by index.

Return A pointer to a transaction trytes, NULL if not found.

Parameters

- `[in] res`: The response object.
- `[in] index`: The index of the list.

size_t **attach_to_tangle_res_trytes_cnt** (*attach_to_tangle_res_t * res*)
Gets the size of the transaction list.

Return The size of the list.

Parameters

- `[in] res`: The response object.

void **attach_to_tangle_res_free** (*attach_to_tangle_res_t ** res*)
Frees an attach to tangle response object.

Parameters

- `[in] res`: the response object.

struct attach_to_tangle_res_t
#include <*attach_to_tangle.h*> The data structure of attach to tangle response.

Public Members

hash8019_array_p **trytes**
Transaction trytes that include a valid nonce field which you can input into *iota_client_broadcast_transactions* and *iota_client_store_transactions*.

The last 243 trytes of the return value consist of the following: **trunk transaction** + **branch transaction** + **nonce**.

Unnamed Group

typedef struct *check_consistency_res_s* **check_consistency_res_t**
The data structure of check consistency response.

*check_consistency_res_t** **check_consistency_res_new**()
Allocates a check consistency response object.

Return A pointer to the response object.

retcode_t **check_consistency_res_info_set** (*check_consistency_res_t* * *res*, char const *const *info*)
Sets the info field of the response object.

Return retcode_t

Parameters

- `[in] res`: The response object.
- `[in] info`: The info string.

void **check_consistency_res_free** (*check_consistency_res_t* ** *res*)
Frees a check consistency response.

Parameters

- `[in] res`: the response object.

struct check_consistency_res_s
#include <*check_consistency.h*> The data structure of check consistency response.

Public Members

bool **state**
State of the given transactions in the tails parameter. A **true** value means that all given transactions are consistent. A **false** value means that one or more of the given transactions aren't consistent.

char_buffer_t* **info**
If the **state** field is **false**, this field contains information about why the transaction is inconsistent.

Unnamed Group

typedef struct *error_res_s* **error_res_t**
The data structure of the error response.

*error_res_t** **error_res_new** (char const *const *error*)
Allocates an error response object.

Return A pointer to the response object.

Parameters

- `[in] error`: The error string for this error response.

char* **error_res_get_message** (*error_res_t* const *const *res*)
Gets the error string.

Return A pointer to the error string.

Parameters

- `[in] res`: The response object.

void **error_res_free** (*error_res_t* ** *res*)
Frees an error response object.

Parameters

- `[in] res`: the response object.

struct error_res_s
#include <error.h> The data structure of the error response.

Public Members

char_buffer_t* **error**
An error string

Unnamed Group

typedef struct *find_transactions_res* **find_transactions_res_t**
The data structure of find transactions response.

*find_transactions_res_t** **find_transactions_res_new**()
Allocates a find transactions response object.

Return A pointer to find transactions response object.

void **find_transactions_res_free**(*find_transactions_res_t* ** *res*)
Frees a find transactions response object.

Parameters

- `[in] res`: The response object.

static *retcode_t* **find_transactions_res_hashes_add**(*find_transactions_res_t* *const *res*, *flex_trit_t* const *const *hash*)
Adds a hash to the response.

Return retcode_t

Parameters

- `[in] res`: The response object.
- `[in] hash`: A hash.

static *flex_trit_t** **find_transactions_res_hashes_get**(*find_transactions_res_t* *const *res*, size_t *index*)
Gets a hash by index.

Return A pointer to the hash. NULL if the index is invalid.

Parameters

- `[in] res`: The response object.
- `[in] index`: The index of hash list.

struct find_transactions_res
#include <find_transactions.h> The data structure of find transactions response.

Public Members

hash243_queue_t **hashes**
The transaction hashes which are returned depend on your input.

- bundles : returns an array of transaction hashes that contain the given bundle hash.
- addresses : returns an array of transaction hashes that contain the given address in the address field.
- tags : returns an array of transaction hashes that contain the given value in the tag field.
- approvees : returns an array of transaction hashes that contain the given transactions in their branch transaction or trunk transaction fields.

Unnamed Group

*get_balances_res_t** **get_balances_res_new**()
Allocates a get balances response object.

Return A pointer to the response object.

void **get_balances_res_free**(*get_balances_res_t* ** *res*)
Frees a get balances response object.

Parameters

- `[in] res`: The response object.

static *retcode_t* **get_balances_res_reference_add**(*get_balances_res_t* *const *res*, *flex_trit_t* const *const *hash*)
Adds a reference hash to the response object.

Return retcode_t

Parameters

- `[in] res`: The response object.
- `[in] hash`: A reference hash.

static *flex_trit_t** **get_balances_res_reference_get**(*get_balances_res_t* *const *res*, size_t *index*)
Gets a reference hash by index.

Return A pointer to a reference hash. NULL if index is invalid.

Parameters

- `[in] res`: The response object.
- `[in] index`: An index of reference list.

size_t **get_balances_res_balances_num**(*get_balances_res_t* const *const *res*)
Gets size of the balances element in the response.

Return The size of the balances.

Parameters

- `[in] res`: The response.

uint64_t **get_balances_res_balances_at**(*get_balances_res_t* const *const *res*, size_t const *index*)
Gets a balance value by index.

Return The balance to the corresponding index.

Parameters

- `[in] res`: The response object.
- `[in] index`: An index of the balance list.

retcode_t **get_balances_res_balances_add**(*get_balances_res_t* *const *res*, uint64_t *value*)
Adds a balance to the response object.

Return retcode_t

Parameters

- `[in] res`: The response object.
- `[in] value`: A balance value.

struct get_balances_res_t
#include <get_balances.h> The data structure of the get balances response.

Public Members

uint64_t **milestone_index**
The index of the milestone that confirmed the most recent balance.

UT_array* **balances**
Array of balances in the same order as the **addresses** parameters were passed to the endpoint.

hash243_queue_t **references**
The referencing tips. If no **tips** parameter was passed to the endpoint, this field contains the hash of the latest milestone that confirmed the balance

Unnamed Group

*get_inclusion_states_res_t** **get_inclusion_states_res_new**()
Allocates a get inclusion states response object.

Return A pointer to the response object.

retcode_t **get_inclusion_states_res_states_add**(*get_inclusion_states_res_t* * *res*, int *state*)
Adds a state to the response object.

Return retcode_t

Parameters

- `[in] res`: The response object.
- `[in] state`: The state of the transaction.

void **get_inclusion_states_res_free**(*get_inclusion_states_res_t* ** *res*)
Frees a get inclusion states response object.

Parameters

- `[in] res`: The response object.

bool **get_inclusion_states_res_states_at**(*get_inclusion_states_res_t* * *res*, size_t *index*)
Gets the transaction state by index.

Return Boolean. **True** means the transaction was confirmed, otherwise **false**.

Parameters

- `[in] res`: The response object.
- `[in] index`: An index of the list.

size_t **get_inclusion_states_res_states_count**(*get_inclusion_states_res_t* * *res*)
Gets the size of the states list.

Return The size of the state list.

Parameters

- `[in] res`: The response object.

struct get_inclusion_states_res_t
#include <*get_inclusion_states.h*> The data structure of the get inclusion states response.

Public Members

UT_array* **states**
List of boolean values in the same order as the **transactions** parameters. A **true** value means the transaction was confirmed.

Unnamed Group

typedef struct *get_missing_transactions_res_s* **get_missing_transactions_res_t**
The data structure of get missing transactions.

*get_missing_transactions_res_t** **get_missing_transactions_res_new**()
Allocates a get missing transactions response.

Return A pointer to the response.

size_t **get_missing_transactions_res_hash_num**(*get_missing_transactions_res_t* * *res*)
Gets the size of the transaction list.

Return The size of the transaction list.

Parameters

- `[in] res`: The response.

void **get_missing_transactions_res_free**(*get_missing_transactions_res_t* ** *res*)
Frees a get missing transactions response object.

Parameters

- `[in] res`: The response object.

static *retcode_t* **get_missing_transactions_res_hashes_add**(*get_missing_transactions_res_t* *const *res*, *flex_trit_t* const *const *hash*)

Adds a transaction hash to the list.

Return retcode_t

Parameters

- `[in] res`: The response object.
- `[in] hash`: A transaction hash.

struct get_missing_transactions_res_s

#include <get_missing_transactions.h> The data structure of get missing transactions.

Public Members

hash243_stack_t **hashes**

List of transaction hashes

Unnamed Group

typedef UT_array **get_neighbors_res_t**

The data structure of get neighbor response.

*get_neighbors_res_t** **get_neighbors_res_new**()

Allocates a get neighbors response object.

Return A pointer to the response object.

void **get_neighbors_res_free**(*get_neighbors_res_t* ** *res*)

Frees a get neighbors response.

Parameters

- `[in] res`: The response object.

static size_t **get_neighbors_res_num**(*get_neighbors_res_t* * *res*)

Gets the number of neighbors in the response.

Return The number of neighbors.

Parameters

- `[in] res`: The response object.

*neighbor_info_t** **get_neighbors_res_neighbor_at**(*get_neighbors_res_t* * *res*, size_t *index*)

Gets a neighbor object by index.

Return A pointer to a neighbor object.

Parameters

- `[in] res`: The response object.

- `[in] index`: An index of the neighbor list.

retcode_t **get_neighbors_res_add_neighbor**(*get_neighbors_res_t* * *res*, char const *const *addr*, uint32_t *all_trans_num*, uint32_t *random_trans_req_num*, uint32_t *new_trans_num*, uint32_t *invalid_trans_num*, uint32_t *stale_trans_num*, uint32_t *sent_trans_num*, char const *const *connection_type*)

Adds a neighbor to the response.

Return retcode_t

Parameters

- `[in] res`: The response object.
- `[in] addr`: The string of an address.
- `[in] all_trans_num`: The value of *neighbor_info_t::all_trans_num*
- `[in] random_trans_req_num`: The value of *neighbor_info_t::random_trans_req_num*
- `[in] new_trans_num`: The value of *neighbor_info_t::new_trans_num*
- `[in] invalid_trans_num`: The value of *neighbor_info_t::invalid_trans_num*
- `[in] stale_trans_num`: The value of *neighbor_info_t::stale_trans_num*
- `[in] sent_trans_num`: The value of *neighbor_info_t::sent_trans_num*
- `[in] connection_type`: A string of **TCP** or **UDP**.

struct neighbor_info_t

#include <get_neighbors.h> The data structure of the neighbor object.

Public Members

char_buffer_t* **address**

address and port number of your peer.

uint32_t **all_trans_num**

Number of all transactions(Invalid, valid and already-seen) this neighbor sent to you.

uint32_t **random_trans_req_num**

Number of random tips this neighbor requested from your node.

uint32_t **new_trans_num**

Number of new transactions received from this neighbor. New transactions are transactions that you have not yet stored when this neighbor sends it to you.

uint32_t **invalid_trans_num**

Number of invalid transactions this neighbor sent to you. These are transactions with invalid signatures or overall schema.

uint32_t **stale_trans_num**

Stale transactions this neighbor has sent you. These are transactions with a timestamp older than your latest snapshot.

uint32_t **sent_trans_num**

Amount of transactions sent through this neighbor. Number of all transactions you get from other neighbors and forward to this neighbor.

char_buffer_t* **connection_type**
The method type this neighbor is using to connect (TCP / UDP).

Unnamed Group

typedef struct *get_node_api_conf_res_s* **get_node_api_conf_res_t**
The data structure of the get node configuration response.

struct get_node_api_conf_res_s
#include <*get_node_api_conf.h*> The data structure of the get node configuration response.

Public Members

uint32_t **max_find_transactions**
The maximum number of transactions that may be returned.

uint32_t **max_requests_list**
The maximum number of parameters in an API call.

uint32_t **max_get_trytes**
The maximum number of trytes that may be returned by the *iota_client_get_trytes*.

uint32_t **max_body_length**
The maximum number of characters that the body of an API call may contain.

uint32_t **milestone_start_index**
The starting milestone index that the node uses to validate and confirm transactions.

bool **test_net**
The type of this node, True runs on testnet else mainnet.

Unnamed Group

typedef struct *get_node_info_res_s* **get_node_info_res_t**
The data structure of the get node info response.

*get_node_info_res_t** **get_node_info_res_new**()
Allocates a get node info response object.

Return A pointer to the response object.

void **get_node_info_res_free**(*get_node_info_res_t* ** *res*)
Frees a get node info response object.

Parameters

- `[in] res`: The response object.

static *retcode_t* **get_node_info_res_app_name_set**(*get_node_info_res_t* *const *res*, char const *const *name*)
Sets application name to the response.

Return retcode_t

Parameters

- `[in] res`: The response object.

- `[in] name`: A string of the application name.

static char const* **get_node_info_res_app_name**(*get_node_info_res_t* const *const *res*)
Gets the application name.

Return A pointer to the application name.

Parameters

- `[in] res`: The response object.

static *retcode_t* **get_node_info_res_app_version_set**(*get_node_info_res_t* *const *res*, char const *const *version*)
Sets the application version.

Return retcode_t

Parameters

- `[in] res`: The response object.
- `[in] version`: A string of the application version.

static char const* **get_node_info_res_app_version**(*get_node_info_res_t* const *const *res*)
Gets the application version.

Return A pointer to the application version.

Parameters

- `[in] res`: The response object.

static *retcode_t* **get_node_info_res_lm_set**(*get_node_info_res_t* *const *res*, *flex_trit_t* const *const *hash*)
Sets the latest milestone hash.

Return retcode_t

Parameters

- `[in] res`: The response object
- `[in] hash`: A milestone hash.

static *flex_trit_t* const* **get_node_info_res_lm**(*get_node_info_res_t* const *const *res*)
Gets the latest milestone hash.

Return A pointer to the hash of latest milestone.

Parameters

- `[in] res`: The response object.

static *retcode_t* **get_node_info_res_lssm_set**(*get_node_info_res_t* *const *res*, *flex_trit_t* const *const *hash*)
Sets the latest solid subtangle milestone hash.

Return retcode_t

Parameters

- `[in] res`: The response object.

- `[in] hash`: A pointer to the hash of latest solid subtangle milestone.

static *flex_trit_t* const* **get_node_info_res_lssm**(*get_node_info_res_t* const *const *res*)
Gets the latest solid subtangle milestone hash.

Return A pointer to the hash of latest solid subtangle milestone.

Parameters

- `[in] res`: The response object.

static *retcode_t* **get_node_info_res_coordinator_address_set**(*get_node_info_res_t* *const *res*, *flex_trit_t* const *const *hash*)
Sets the coordinator address.

Return retcode_t

Parameters

- `[in] res`: The response object.
- `[in] hash`: A coordinator hash.

static *flex_trit_t* const* **get_node_info_res_coordinator_address**(*get_node_info_res_t* const *const *res*)

static char const* **get_node_info_res_features_at**(*get_node_info_res_t* * *res*, size_t *idx*)

static size_t **get_node_info_req_features_len**(*get_node_info_res_t* * *res*)

struct get_node_info_res_s
#include <get_node_info.h> The data structure of the get node info response.

Public Members

char_buffer_t* **app_name**
Name of the IOTA software you're currently using.

char_buffer_t* **app_version**
The version of the IOTA software you're currently running.

uint64_t **time**
Current UNIX timestamp.

uint32_t **latest_milestone_index**
Index of the latest milestone.

uint32_t **latest_solid_subtangle_milestone_index**
Index of the latest solid subtangle.

uint32_t **milestone_start_index**
The start index of the milestones. This index is encoded in each milestone transaction by the coordinator

uint32_t **tips**
Number of tips in the network.

uint32_t **transactions_to_request**
Transactions to request during syncing process.

uint16_t **neighbors**
Number of neighbors you are directly connected with.

uint16_t **packets_queue_size**
Packets which are currently queued up.

flex_trit_t get_node_info_res_s::latest_milestone[FLEX_TRIT_SIZE_243]
The hash of the latest transaction that was signed off by the coordinator.

flex_trit_t get_node_info_res_s::latest_solid_subtangle_milestone[FLEX_TRIT_SIZE_243]
The hash of the latest transaction which is solid and is used for sending transactions. For a milestone to become solid, your local node must approve the subtangle of coordinator-approved transactions, and have a consistent view of all referenced transactions.

UT_array* **features**

flex_trit_t get_node_info_res_s::coordinator_address[FLEX_TRIT_SIZE_243]
The address of the coordinator being followed by this node.

Unnamed Group

typedef struct *get_tips_res_s* **get_tips_res_t**
The data structure of the get tips response.

*get_tips_res_t** **get_tips_res_new**()
Allocates a get tips response.

Return A pointer to the response object.

size_t **get_tips_res_hash_num**(*get_tips_res_t* * *res*)
Gets the number of tip transactions.

Return Number of tip transactions.

Parameters

- `[in] res`: The response object.

void **get_tips_res_free**(*get_tips_res_t* ** *res*)
Frees a get tips response.

Parameters

- `[in] res`: The response object.

static *retcode_t* **get_tips_res_hashes_push**(*get_tips_res_t* * *res*, *flex_trit_t* const *const *hash*)
Adds a tip transaction hash to the response.

Return retcode_t

Parameters

- `[in] res`: The response object.
- `[in] hash`: A tip transaction hash.

static *retcode_t* **get_tips_res_hashes_pop**(*get_tips_res_t* * *res*, *flex_trit_t* *const *buf*)
Removes a tip transaction from the response.

Return retcode_t

Parameters

- `[in] res`: The response object.
- `[in] buf`: A tip transaction hash.

struct get_tips_res_s
#include <get_tips.h> The data structure of the get tips response.

Public Members

hash243_stack_t **hashes**
Current tip transaction hashes

Unnamed Group

typedef struct *get_transactions_to_approve_res_s* **get_transactions_to_approve_res_t**
The data structure of the get transactions to approve response.

*get_transactions_to_approve_res_t** **get_transactions_to_approve_res_new**()
Allocates a get transactions to approve response.

Return The response object.

void **get_transactions_to_approve_res_free**(*get_transactions_to_approve_res_t* **const *res*)
Frees a get transactions to approve response.

Parameters

- `[in] res`: The response object.

static *retcode_t* **get_transactions_to_approve_res_set_branch**(*get_transactions_to_approve_res_t* *const *res*, *flex_trit_t* const *const *branch*)
Sets the branch transaction to the response.

Return retcode_t

Parameters

- `[in] res`: The response object.
- `[in] branch`: A valid branch transaction hash.

static *flex_trit_t* const* **get_transactions_to_approve_res_branch**(*get_transactions_to_approve_res_t* const *const *res*)
Gets the branch transaction from the response.

Return A pointer to the branch transaction hash.

Parameters

- `[in] res`: The response object.

static *retcode_t* **get_transactions_to_approve_res_set_trunk**(*get_transactions_to_approve_res_t* *const *res*, *flex_trit_t* const *const *trunk*)
Sets the trunk transaction to the response.

Return retcode_t

Parameters

- `[in] res`: The response object.
- `[in] trunk`: A valid trunk transaction hash.

static *flex_trit_t* const* **get_transactions_to_approve_res_trunk**(*get_transactions_to_approve_res_t* const *const *res*)

Gets the trunk transaction from the response.

Return A pointer to the trunk transaction hash.

Parameters

- `[in] res`: The response object.

struct get_transactions_to_approve_res_s

#include <*get_transactions_to_approve.h*> The data structure of the get transactions to approve response.

Public Members

flex_trit_t get_transactions_to_approve_res_s::branch[FLEX_TRIT_SIZE_243]

The valid branch transaction hash

flex_trit_t get_transactions_to_approve_res_s::trunk[FLEX_TRIT_SIZE_243]

The valid trunk transaction hash

Unnamed Group

typedef struct *get_trytes_res_s* **get_trytes_res_t**

The data structure of the get trytes response.

*get_trytes_res_t** **get_trytes_res_new**()

Allocates a get trytes response.

Return A pointer to the response object.

void **get_trytes_res_free**(*get_trytes_res_t* **const *res*)

Frees a get trytes response.

Parameters

- `[in] res`: The response object.

static *retcode_t* **get_trytes_res_trytes_add**(*get_trytes_res_t* *const *res*, *flex_trit_t* const *const *hash*)

Adds a raw transaction to the response.

Return retcode_t

Parameters

- `[in] res`: The response object.
- `[in] hash`: A raw transaction hash.

static *flex_trit_t** **get_trytes_res_trytes_get** (*get_trytes_res_t* *const *res*, size_t *index*)
Gets a raw transaction by index.

Return A pointer to a raw transaction hash.

Parameters

- `[in] res`: The response object.
- `[in] index`: An index of the transaction list.

struct get_trytes_res_s
#include <get_trytes.h> The data structure of the get trytes response.

Public Members

hash8019_queue_t **trytes**

Unnamed Group

typedef struct *remove_neighbors_res_s* **remove_neighbors_res_t**
The data structure of the remove neighbors response.

*remove_neighbors_res_t** **remove_neighbors_res_new** ()
Allocates a remove neighbors response.

Return A pointer to the response object.

void **remove_neighbors_res_free** (*remove_neighbors_res_t* ** *res*)
Frees a remove neighbors response.

Parameters

- `[in] res`: The response object.

struct remove_neighbors_res_s
#include <remove_neighbors.h> The data structure of the remove neighbors response.

Public Members

int **removed_neighbors**
Total number of removed neighbors

Unnamed Group

typedef struct *were_addresses_spent_from_res_s* **were_addresses_spent_from_res_t**
The data structure of were addresses spent from response.

*were_addresses_spent_from_res_t** **were_addresses_spent_from_res_new** ()
Allocates a were addresses spent from response object.

Return A pointer to the response object.

retcode_t **were_addresses_spent_from_res_states_add** (*were_addresses_spent_from_res_t* * *res*, int *state*)

Adds a state to the response object.

Return retcode_t

Parameters

- `[in] res`: The response object.
- `[in] state`: The state of the address.

void **were_addresses_spent_from_res_free** (*were_addresses_spent_from_res_t* ** *res*)

Frees a were addresses spent from response.

Parameters

- `[in] res`: the response object.

bool **were_addresses_spent_from_res_states_at** (*were_addresses_spent_from_res_t* * *res*, size_t *index*)

Gets the address state by index.

Return Boolean. **True** means the address was spent, otherwise **false**.

Parameters

- `[in] res`: The response object.
- `[in] index`: An index of the list.

size_t **were_addresses_spent_from_res_states_count** (*were_addresses_spent_from_res_t* * *res*)

Gets the size of the states list.

Return The size of the state list.

Parameters

- `[in] res`: The response object.

struct were_addresses_spent_from_res_s

#include <*were_addresses_spent_from.h*> The data structure of were addresses spent from response.

Public Members

UT_array* **states**

States of the given addresses in the parameter. A **true** value means that the given address was spent. A **false** value means that the given address has not been spent.

CHAPTER 2

IOTA Common Libraries

The common libraries are including Crypto, Helpers, Models, and Trinary components.

2.1 Common

typedef enum *retcode_t* **retcode_t**

Error codes, see error.h

const char* **error_2_string**(*retcode_t err*)
error code to string

Return string

Parameters

- `err`: error code

2.2 Crypto

TODO

2.3 Helpers

TODO

2.4 Models

Unnamed Group

MAX_IOTA_SUPPLY

BUNDLE_FOREACH(txs, tx)
The bundle iterator.

bundle_status_e
bundle validation status.

Values:

0

BUNDLE_VALID

BUNDLE_EMPTY

BUNDLE_INCOMPLETE

BUNDLE_INVALID_TX

BUNDLE_INVALID_INPUT_ADDRESS

BUNDLE_INVALID_VALUE

BUNDLE_INVALID_HASH

BUNDLE_INVALID_SIGNATURE

typedef enum *bundle_status_e* **bundle_status_t**
bundle validation status.

typedef UT_array **bundle_transactions_t**

typedef UT_array **bundle_hashes_t**

void **bundle_transactions_new**(*bundle_transactions_t* **const *bundle*)
Allocates a bundle object.

Parameters

- `[in] bundle`: A bundle object.

void **bundle_transactions_free**(*bundle_transactions_t* **const *bundle*)
Frees a bundle object.

Parameters

- `[in] bundle`: A bundle object.

void **bundle_transactions_add**(*bundle_transactions_t* *const *bundle*, iota_transaction_t const *const *transaction*)
Adds a transaction to the bundle.

Parameters

- `[in] bundle`: The bundle object.
- `[in] transaction`: A transaction object.

iota_transaction_t* **bundle_at** (*bundle_transactions_t* *const *bundle*, size_t *index*)
Gets a transaction from the bundle by index.

Return #iota_transaction_t

Parameters

- `[in] bundle`: The bundle object.
- `[in] index`: The index of a transaction.

static size_t **bundle_transactions_size** (*bundle_transactions_t* const *const *bundle*)
Gets the number of transactions in the bundle.

Return An number of transactions.

Parameters

- `[in] bundle`: The bundle object.

static *flex_trit_t* const* **bundle_transactions_bundle_hash** (*bundle_transactions_t* const *const *bundle*)
Get bundle hash from a bundle transaction object.

Return The bundle hash

Parameters

- `[in] bundle`: The bundle transaction object

void **bundle_calculate_hash** (*bundle_transactions_t* * *bundle*, Kerl *const *kerl*, *flex_trit_t* * *out*)
Calculates bundle hash using Kerl.

Parameters

- `[in] bundle`: A bundle object.
- `[in] kerl`: A Kerl object.
- `[out] out`: The bundle hash.

void **bundle_finalize** (*bundle_transactions_t* * *bundle*, Kerl *const *kerl*)
Finalizes a bundle by calculating the bundle hash.

Parameters

- `[in] bundle`: A bundle object.
- `[in] kerl`: A Kerl object.

retcode_t **bundle_validate** (*bundle_transactions_t* *const *bundle*, *bundle_status_t* *const *status*)
Validates a bundle.

Return retcode_t

Parameters

- `[in] bundle`: A bundle object.
- `[out] status`: The status of the bundle.

void **bundle_reset_indexes**(*bundle_transactions_t* *const *bundle*)
Re-calculates the current and last index of the bundle.

Parameters

- `[in] bundle`: A bundle object.

void **bundle_set_messages**(*bundle_transactions_t* * *bundle*, signature_fragments_t * *messages*)
Sets bundle message.

Parameters

- `[in] bundle`: A bundle object.
- `[in] messages`: A message for the bundle.

retcode_t **bundle_sign**(*bundle_transactions_t* *const *bundle*, *flex_trit_t* const *const *seed*, *inputs_t* const *const *inputs*, Kerl *const *kerl*)
Adds signature to transactions in a bundle.

Return retcode_t

Parameters

- `[in] bundle`: A bundle object.
- `[in] seed`: The seed of inputs.
- `[in] inputs`: The inputs for the bundle.
- `[in] kerl`: A Kerl object.

Unnamed Group

INPUTS_FOREACH(inputs, elm)
loop over an input list

typedef UT_array **input_array_t**

UT_icd const input_array_icd = {sizeof(input_t), NULL, NULL, NULL}

static *retcode_t* **inputs_append**(*inputs_t* * *inputs*, *input_t* * *input*)
append an input object to the input list

Return retcode_t

Parameters

- `inputs`: the input list
- `input`: an input object for appending

static void **inputs_clear**(*inputs_t* * *inputs*)
clear all elements

Parameters

- `inputs`:

static int64_t **inputs_balance**(*inputs_t* * *inputs*)
total balance in this input list

Return int64_t

Parameters

- `inputs`:

static size_t **inputs_len** (*inputs_t* const *const *inputs*)
size of the input list

Return size_t

Parameters

- `inputs`:

static *input_t** **inputs_at** (*inputs_t* * *inputs*, size_t *index*)
get an input object by index

Return input_t*

Parameters

- `inputs`:
- `index`:

struct input_t
#include <*inputs.h*> input object

Public Members

int64_t **balance**

uint64_t **key_index**

uint8_t **security**

flex_trit_t input_t::address[FLEX_TRIT_SIZE_243]

struct inputs_t
#include <*inputs.h*> a list of input object

Public Members

int64_t **total_balance**

*input_array_t** **input_array**

2.5 Trinary

typedef int8_t **flex_trit_t**

static trit_t **flex_trits_at** (*flex_trit_t* const *const *flex_trits*, size_t const *len*, size_t *index*)
Returns the trit at a given index in an array of flex_trits

Return trit_t - the trit at the given index

Parameters

- `[in] flex_trits`: - an array of flex_trits
- `[in] len`: - the number of trits encoded in the flex_trit array
- `[in] index`: - the index of the trit to access

static uint8_t **flex_trits_set_at** (*flex_trit_t* *const *flex_trits*, size_t const *len*, size_t *index*, trit_t *trit*)

Set the trit at a given index in an array of trits to the given value

Parameters

- `[in] flex_trits`: - an array of flex_trits
- `[in] len`: - the number of trits encoded in the flex_trit array
- `[in] index`: - the index of the trit to access
- `[in] trit`: - the trit value to set

static bool **flex_trits_are_null** (*flex_trit_t* const *const *flex_trits*, size_t const *len*)

size_t **flex_trits_slice** (*flex_trit_t* *const *to_flex_trits*, size_t const *to_len*, *flex_trit_t* const *const *flex_trits*, size_t const *len*, size_t const *start*, size_t const *num_trits*)

Returns a portion of length num_trits of an array into a new array from start. The original array will not be modified.

Return size_t - the number of trits extracted

Parameters

- `[in] to_flex_trits`: - the array that will contain the slice
- `[in] to_len`: - the number of trits encoded in the to_flex_trits array
- `[in] flex_trits`: - the original array
- `[in] len`: - the number of trits the flex_trits array stores
- `[in] start`: - the start index in the original array
- `[in] num_trits`: - the number of trits to extract

size_t **flex_trits_insert** (*flex_trit_t* *const *to_flex_trits*, size_t const *to_len*, *flex_trit_t* const *const *flex_trits*, size_t const *len*, size_t const *start*, size_t const *num_trits*)

Inserts the contents of an array into another array starting at a given index.

Return size_t - the number of trits copied over

Parameters

- `[in] to_flex_trits`: - the array to insert into
- `[in] to_len`: - the number of trits encoded in the to_flex_trits array
- `[in] flex_trits`: - the array containing the trits to copy over
- `[in] len`: - the number of trits the flex_trits array stores
- `[in] start`: - the start index in the destination array
- `[in] num_trits`: - the number of trits to copy over

size_t **flex_trits_to_trits** (trit_t *const *trits*, size_t const *to_len*, *flex_trit_t* const *const *flex_trits*, size_t const *len*, size_t const *num_trits*)

Returns an array of trits regardless of the current memory storage scheme

Return size_t - the number of trits encoded

Parameters

- `[in] trits`: - an array of individual trits
- `[in] to_len`: - the number of trits the trits array contains
- `[in] flex_trits`: - the array of packed trits
- `[in] len`: - the number of trits the flex_trits array stores
- `[in] num_trits`: - the number of trits to extract

size_t **flex_trits_from_trits** (*flex_trit_t* *const *to_flex_trits*, size_t const *to_len*, trit_t const *const *trits*, size_t const *len*, size_t const *num_trits*)

Returns an array of trits packed in the current memory storage scheme

Return size_t - the number of trits decoded

Parameters

- `[in] to_flex_trits`: - an array of flex_trits
- `[in] to_len`: - the number of trits encoded in the to_flex_trits array
- `[in] trits`: - an array of individual trits
- `[in] len`: - the number of trits the trits array contains
- `[in] num_trits`: - the number of trits to pack

size_t **flex_trits_from_trytes** (*flex_trit_t* * *to_flex_trits*, size_t *to_len*, const tryte_t * *trytes*, size_t *len*, size_t *num_trytes*)

Returns an array of flex_trits.

Return size_t - the number of trytes decoded

Parameters

- `[in] to_flex_trits`: - the array of packed trits
- `[in] to_len`: - the number of trits in the to_flex_trits array
- `[in] trytes`: - an array of trytes
- `[in] len`: - the number of trytes in the trytes array
- `[in] num_trytes`: - the size of trytes to unpack

size_t **flex_trits_to_bytes** (byte_t * *bytes*, size_t *to_len*, const *flex_trit_t* * *flex_trits*, size_t *len*, size_t *num_trits*)

Returns an array of bytes.

Return size_t - the number of trits encoded

Parameters

- `[in] bytes`: - an array to store bytes
- `[in] to_len`: - the number of trits the bytes array contains
- `[in] flex_trits`: - the array of packed trits
- `[in] len`: - the number of trits the flex_trits array contains
- `[in] num_trits`: - the number of trits to pack

size_t **flex_trits_from_bytes** (*flex_trit_t* * *to_flex_trits*, size_t *to_len*, const byte_t * *bytes*, size_t *len*, size_t *num_trits*)

Returns an array of flex_trits.

Return size_t - the number of trits decoded

Parameters

- `[in] to_flex_trits`: - the array of packed trits
- `[in] to_len`: - the number of trits in the to_flex_trits array
- `[in] bytes`: - an array of bytes
- `[in] len`: - the number of trits in the bytes array
- `[in] num_trits`: - the number of trits to unpack

void **`flex_trit_print`** (*flex_trit_t* const * *trits*, size_t *trits_len*)

Print flex trits through stdout.

Parameters

- `trits`: A pointer to flex trits
- `trits_len`: Number of trit

Index

G

I

R

S

W