# **BBc1 Documentation**

Release 1.0

beyond-blockchain.org

# Contents:

1	bbc1	package														1
	1.1	Subpackages .						 	 	 		 				1
		1.1.1 bbc1.cc	ore package					 	 	 		 				1
		1.1.1.1	Subpackages					 	 	 		 				1
		1.1.1.2	Submodules					 	 	 		 				2
		1.1.1.3	Module contents					 	 	 		 				43
	1.2	Module contents	3					 	 	 		 				43
2 Indices and tables									45							
Python Module Index							47									
In	ndex 4							49								

# CHAPTER 1

bbc1 package

## 1.1 Subpackages

## 1.1.1 bbc1.core package

## 1.1.1.1 Subpackages

bbc1.core.libbbcsig package

## **Submodules**

## bbc1.core.libbbcsig.pybbcsig module

```
class bbc1.core.libbbcsig.pybbcsig.PyBBcSigSecp256k1
    Bases: object
    convert_from_der(der_data, pubkey_type)
    convert_from_pem(pem_data, pubkey_type)
    generate_keypair(pubkey_type)
    get_public_key_compressed(privkey)
    get_public_key_uncompressed(privkey)
    output_der(privkey)
    output_pem(privkey)
    sign(privkey, digest)
    verify(pubkey, digest, signature)
```

### **Module contents**

### 1.1.1.2 Submodules

## bbc1.core.bbc\_app module

Copyright (c) 2017 beyond-blockchain.org.

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

Bases: object

Basic functions for a client of bbc\_core

```
cancel_insert_completion_notification(asset_group_id)
```

Cancel notification when a transaction has been inserted (as a copy of transaction)

**Parameters** asset\_group\_id (bytes) - asset\_group\_id for requesting notification about insertion

Returns query\_id

Return type bytes

domain\_close (domain\_id=None)

Close domain leading to remove\_domain in the core

Parameters domain\_id (bytes) - domain\_id to delete

Returns query\_id

Return type bytes

domain\_setup (domain\_id, config=None)

Set up domain with the specified network module and storage

This method should be used by a system administrator.

#### **Parameters**

- domain\_id (bytes) domain\_id to create
- **config** (str) system config in json format

Returns query\_id

Return type bytes

#### exchange\_key()

Perform ECDH (key exchange algorithm)

Returns query\_id

Return type bytes

```
gather_signatures (txobj, reference_obj=None, asset_files=None, destinations=None, any-
                        cast=False)
     Request to gather signatures from the specified user_ids
         Parameters
             • txobj (BBcTransaction) -
             • reference_obj (BBcReference) - BBcReference object that includes the informa-
               tion about destinations
             • asset_files (dict) - mapping from asset_id to its file content
             • destinations (list) - list of destination user_ids
             • anycast (bool) - True if this message is for anycasting
         Returns query_id
         Return type bytes
get_bbc_config()
     Get config file of bbc_core
     This method should be used by a system administrator.
         Returns query_id
         Return type bytes
get_domain_list()
     Get domain_id list in bbc_core
         Returns query_id
         Return type bytes
get_domain_neighborlist(domain_id)
     Get peer list of the domain from the core node
     This method should be used by a system administrator.
         Parameters domain_id (bytes) - domain_id of the neighbor list
         Returns query_id
         Return type bytes
get_forwarding_list()
     Get forwarding list of the domain in the core node
         Returns query_id
         Return type bytes
get_node_id()
     Get node_id of the connecting core node
         Returns query_id
         Return type bytes
get_notification_list()
     Get notification list of the core node
         Returns query_id
         Return type bytes
```

```
get_stats()
     Get statistics of bbc core
         Returns query_id
         Return type bytes
get user list()
     Get user_ids in the domain that are connecting to the core node
         Returns query_id
         Return type bytes
include_admin_info (dat, admin_info, keypair)
include_cross_ref(txobj)
     Include BBcCrossRef from other domains in the transaction
     If the client object has one or more cross_ref objects, one of them is included in the given transaction. This
     method should be voluntarily called for inter-domain weak collaboration.
         Parameters txobj (BBcTransaction) - Transaction object to include cross_ref
insert transaction (tx obj)
     Request to insert a legitimate transaction
         Parameters tx_obj (BBcTransaction) - Transaction object to insert
         Returns query id
         Return type bytes
manipulate_ledger_subsystem(enable=False, domain_id=None)
     Start/stop ledger_subsystem on the bbc_core
     This method should be used by a system administrator.
         Parameters
             • enable (bool) - True->start, False->stop
             • domain_id (bytes) - target domain_id to enable/disable ledger_subsystem
         Returns query_id
         Return type bytes
notify_domain_key_update()
     Notify update of bbc_core
     This method should be used by a system administrator.
         Returns query_id
         Return type bytes
receiver_loop()
register_in_ledger_subsystem(asset_group_id, transaction_id)
     Register transaction_id in the ledger_subsystem
         Parameters
             • asset_group_id(bytes)-
             • transaction_id (bytes) - the target transaction_id
         Returns query_id
```

```
Return type bytes
register_to_core (on_multiple_nodes=False)
     Register the client (user_id) to the core node
     After that, the client can communicate with the core node.
         Parameters on multiple nodes (bool) – True if this user id is for multicast address
         Returns True
         Return type bool
request_cross_ref_holders_list()
     Request the list of transaction_ids that are registered as cross_ref in outer domains
         Returns query_id
         Return type bytes
request_insert_completion_notification(asset_group_id)
     Request notification when a transaction has been inserted (as a copy of transaction)
         Parameters asset_group_id (bytes) - asset_group_id for requesting notification about
             insertion
         Returns query_id
         Return type bytes
request_to_repair_asset (asset_group_id, asset_id)
     Request to repair compromised asset file
         Parameters
             • asset_group_id (bytes) - the asset_group_id of the target asset
             • asset_id (bytes) - the target asset_id
         Returns query_id
         Return type bytes
request_to_repair_transaction(transaction_id)
     Request to repair compromised transaction data
         Parameters transaction_id (bytes) – the target transaction to repair
         Returns query_id
         Return type bytes
request_verify_by_cross_ref(transaction_id)
     Request to verify the transaction by Cross_ref in transaction of outer domain
         Parameters transaction_id (bytes) – the target transaction_id
         Returns query_id
         Return type bytes
search_transaction (transaction_id)
     Search request for a transaction
         Parameters transaction_id (bytes) – the target transaction to retrieve
         Returns query id
```

**Return type** bytes

```
\verb|search_transaction_with_condition|| (asset\_group\_id=None,
```

asset id=None,

user\_id=None, count=1)

Search transaction data by asset\_group\_id/asset\_id/user\_id

If multiple conditions are specified, they are considered as AND condition.

### **Parameters**

- asset\_group\_id (bytes) asset\_group\_id in BBcEvent and BBcRelations
- asset id (bytes) asset id in BBcAsset
- user\_id (bytes) user\_id in BBcAsset that means the owner of the asset
- **count** (*int*) the number of transactions to retrieve

Returns query\_id

Return type bytes

send\_domain\_ping (domain\_id, ipv4=None, ipv6=None, port=6641)

Send domain ping to notify the existence of the node

This method should be used by a system administrator.

## **Parameters**

- domain\_id (bytes) target domain\_id to send ping
- ipv4 (str) IPv4 address of the node
- ipv6 (str) IPv6 address of the node
- port (int) Port number to wait messages UDP

Returns query\_id

Return type bytes

send\_message (msg, dst\_user\_id, is\_anycast=False)

Send a message to the specified user\_id

#### **Parameters**

- msg (dict) message to send
- **dst\_user\_id** (bytes) destination user\_id
- **is\_anycast** (bool) If true, the message is treated as an anycast message.

Returns query\_id

Return type bytes

**sendback\_denial\_of\_sign** (dest\_user\_id=None, transaction\_id=None, reason\_text=None, query\_id=None)

Send back the denial of sign the transaction

This method is called if the receiver (signer) denies the transaction.

## **Parameters**

- **dest\_user\_id** (bytes) destination user\_id to send back
- transaction\_id(bytes)-
- reason\_text (str) message to the requester about why the node denies the transaction
- query\_id The query\_id that was in the received SIGN\_REQUEST message

```
Returns query_id
```

Return type bytes

Send back the signed transaction to the source

This method is called if the receiver (signer) approves the transaction.

## **Parameters**

- **dest\_user\_id** (bytes) destination user\_id to send back
- transaction\_id(bytes)-
- ref\_index (int) (optional) which reference in transaction the signature is for
- **signature** (BBcSignature) **Signature** that expresses approval of the transaction with transaction\_id
- query\_id The query\_id that was in the received SIGN\_REQUEST message

Returns query\_id

Return type bytes

```
set_callback (callback_obj)
```

Set callback object that implements message processing functions

**Parameters** callback\_obj (obj) - callback method object

```
set_domain_id (domain_id)
```

Set domain\_id to this client to include it in all messages

**Parameters domain\_id** (bytes) – domain\_id to join in

```
set_domain_static_node (domain_id, node_id, ipv4, ipv6, port)
```

Set static node to the core node

IPv6 is used for socket communication if both IPv4 and IPv6 is specified. This method should be used by a system administrator.

## **Parameters**

- domain\_id (bytes) target domain\_id to set static neighbor entry
- **node\_id** (bytes) node\_id to register
- ipv4 (str) IPv4 address of the node
- ipv6 (str) IPv6 address of the node
- port (int) Port number to wait messages (UDP/TCP)

**Returns** query\_id

**Return type** bytes

```
set_keypair (keypair)
```

Set keypair for the user

Parameters keypair (KeyPair) - KeyPair object for signing

set\_node\_key (pem\_file=None)

Set node\_key to this client

**Parameters** pem\_file (str) – path string for the pem file

```
set_user_id (identifier)
          Set user_id of the object
              Parameters identifier (bytes) – user_id of this clients
     start_receiver_loop()
     traverse transactions (transaction id, direction=1, hop count=3)
          Search request for transactions
          The method traverses the transaction graph in the ledger. The response from the bbc_core includes the list
          of transactions.
              Parameters
                  • transaction_id (bytes) – the target transaction to retrieve
                  • direction (int) - 1:backforward, non-1:forward
                  • hop_count (int) - hop count to traverse from the specified origin point
              Returns query_id
              Return type bytes
     unregister_from_core()
          Unregister and disconnect from the core node
              Returns True
              Return type bool
     verify_in_ledger_subsystem(asset_group_id, transaction_id)
          Verify transaction_id in the ledger_subsystem
              Parameters
                  • asset_group_id(bytes)-
                  • transaction_id (bytes) - the target transaction_id
              Returns query_id
              Return type bytes
class bbc1.core.bbc_app.Callback(log=None)
     Bases: object
     Set of callback functions for processing received message
     If you want to implement your own way to process messages, inherit this class.
     create_queue (query_id)
     dispatch (dat, payload_type)
     get_from_queue (query_id, timeout=None, no_delete=False)
     proc_cmd_sign_request (dat)
          Callback for message REQUEST_SIGNATURE
          This method should be overridden if you want to process the message asynchronously.
              Parameters dat (dict) – received message
     proc_notify_cross_ref (dat)
          Callback for message NOTIFY CROSS REF
```

This method must not be overridden.

**Parameters dat** (dict) – received message

## proc\_notify\_inserted(dat)

Callback for message NOTIFY INSERTED

This method should be overridden if you want to process the message asynchronously.

**Parameters dat** (dict) – received message

## proc\_resp\_cross\_ref\_list (dat)

Callback for message RESPONSE\_CROSS\_REF\_LIST

This method should be overridden if you want to process the message asynchronously.

Parameters dat (dict) - received message

#### proc\_resp\_domain\_close (dat)

Callback for message RESPONSE\_CLOSE\_DOMAIN

This method should be overridden if you want to process the message asynchronously.

**Parameters dat** (dict) – received message

### proc\_resp\_domain\_setup(dat)

Callback for message RESPONSE\_SETUP\_DOMAIN

This method should be overridden if you want to process the message asynchronously.

**Parameters dat** (dict) – received message

## proc\_resp\_ecdh\_key\_exchange (dat)

Callback for message RESPONSE ECDH KEY EXCHANGE

This method must not be overridden.

**Parameters dat** (dict) – received message

## proc\_resp\_gather\_signature(dat)

Callback for message RESPONSE\_GATHER\_SIGNATURE

This method should be overridden if you want to process the message asynchronously.

**Parameters dat** (dict) – received message

## proc\_resp\_get\_config(dat)

Callback for message RESPONSE\_GET\_CONFIG

This method should be overridden if you want to process the message asynchronously.

**Parameters dat** (dict) – received message

## proc\_resp\_get\_domainlist(dat)

Callback for message RESPONSE\_GET\_DOMAINLIST

List of domain\_ids is queued rather than message itself. This method should be overridden if you want to process the message asynchronously.

Parameters dat (dict) - received message

### proc\_resp\_get\_forwardinglist(dat)

Callback for message RESPONSE\_GET\_FORWARDING\_LIST

List of user\_ids in other core nodes is queued rather than message itself. This method should be overridden if you want to process the message asynchronously.

**Parameters dat** (dict) – received message

## proc\_resp\_get\_neighborlist(dat)

Callback for message RESPONSE\_GET\_NEIGHBORLIST

List of neighbor node info (the first one is that of the connecting core) is queued rather than message itself. This method must not be overridden.

**Parameters dat** (dict) – received message

## proc\_resp\_get\_node\_id(dat)

Callback for message RESPONSE\_GET\_NODEID

Node\_id is queued rather than message itself. This method should be overridden if you want to process the message asynchronously.

**Parameters dat** (dict) – received message

## proc\_resp\_get\_notificationlist(dat)

Callback for message RESPONSE\_GET\_NOTIFICATION\_LIST

List of user\_ids in other core nodes is queued rather than message itself. This method should be overridden if you want to process the message asynchronously.

**Parameters dat** (dict) – received message

## proc\_resp\_get\_stats(dat)

Callback for message RESPONSE\_GET\_STATS

This method should be overridden if you want to process the message asynchronously.

Parameters dat (dict) - received message

## proc\_resp\_get\_userlist(dat)

Callback for message RESPONSE\_GET\_USERS

List of user\_ids is queued rather than message itself. This method should be overridden if you want to process the message asynchronously.

Parameters dat (dict) - received message

## proc\_resp\_insert (dat)

Callback for message RESPONSE\_INSERT

This method should be overridden if you want to process the message asynchronously.

Parameters dat (dict) - received message

## proc\_resp\_ledger\_subsystem(dat)

Callback for message RESPONSE\_MANIP\_LEDGER\_SUBSYS

This method should be overridden if you want to process the message asynchronously.

**Parameters dat** (dict) – received message

## proc\_resp\_register\_hash(dat)

Callback for message RESPONSE\_REGISTER\_HASH\_IN\_SUBSYS

This method should be overridden if you want to process the message asynchronously.

**Parameters dat** (dict) – received message

## proc\_resp\_search\_transaction(dat)

Callback for message RESPONSE\_SEARCH\_TRANSACTION

This method should be overridden if you want to process the message asynchronously.

Parameters dat (dict) - received message

## proc\_resp\_search\_with\_condition(dat)

Callback for message RESPONSE\_SEARCH\_WITH\_CONDITIONS

This method should be overridden if you want to process the message asynchronously.

**Parameters dat** (dict) – received message

#### proc\_resp\_set\_neighbor(dat)

Callback for message RESPONSE\_SET\_STATIC\_NODE

This method should be overridden if you want to process the message asynchronously.

**Parameters dat** (dict) – received message

#### proc\_resp\_sign\_request (dat)

Callback for message RESPONSE\_SIGNATURE

This method should be overridden if you want to process the message asynchronously.

Parameters dat (dict) - received message

## proc\_resp\_traverse\_transactions (dat)

Callback for message RESPONSE\_TRAVERSE\_TRANSACTIONS

This method should be overridden if you want to process the message asynchronously.

Parameters dat (dict) - received message

### proc\_resp\_verify\_cross\_ref (dat)

Callback for message RESPONSE\_CROSS\_REF\_VERIFY

This method should be overridden if you want to process the message asynchronously.

Parameters dat (dict) - received message

## proc\_resp\_verify\_hash(dat)

Callback for message RESPONSE\_VERIFY\_HASH\_IN\_SUBSYS

This method should be overridden if you want to process the message asynchronously.

**Parameters dat** (dict) – received message

## proc\_user\_message (dat)

Callback for message MESSAGE

This method should be overridden if you want to process the message asynchronously.

Parameters dat (dict) - received message

```
set_client(client)
```

set logger (log)

## sync\_by\_queryid (query\_id, timeout=None, no\_delete\_q=False)

Wait for the message with specified query\_id

This method creates a queue for the query\_id and waits for the response

#### **Parameters**

- query\_id (byte) timeout for waiting a message in seconds
- timeout (int) timeout for waiting a message in seconds
- no\_delete\_q (bool) If True, the queue for the query\_id remains after popping a message

**Returns** a received message

## **Return type** dict

```
synchronize(timeout=None)
```

Wait for receiving message with a common queue

**Parameters** timeout (int) – timeout for waiting a message in seconds

Returns a received message

Return type dict

## bbc1.core.bbc\_config module

Copyright (c) 2017 beyond-blockchain.org.

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

```
http://www.apache.org/licenses/LICENSE-2.0
```

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

```
class bbc1.core.bbc_config.BBcConfig(directory=None, file=None)
     Bases: object
     System configuration
     get_config()
          Return config dictionary
     get_domain_config (domain_id, create_if_new=False)
          Return the part of specified domain_id in the config dictionary
     get_json_config()
          Get config in json format
     read_config()
          Read config file
     remove_domain_config(domain_id)
          Remove the part of specified domain id in the config dictionary
     update_config()
          Write config to file (config.json)
bbc1.core.bbc config.update deep (d, u)
     Utility for updating nested dictionary
bbc1.core.bbc core module
exec python "$0" "$@"
```

```
class bbc1.core.bbc_core.BBcCoreService (p2p_port=None, use_domain0=False, ip4addr=None, ip6addr=None, workingdir='.bbc1', configfile=None, use_nodekey=None, use_ledger_subsystem=False, loglevel='all', logname='-', server_start=True)

Base service object of BBc-1
```

insert\_transaction (domain\_id, txdata, asset\_files)

Insert transaction into ledger

### **Parameters**

- domain\_id (bytes) target domain\_id
- txdata (bytes) serialized transaction data
- asset\_files (dict) dictionary of {asset\_id: content} for the transaction

Returns inserted transaction\_id or error message

Return type dictlstr

```
quit_program()
```

Processes when quiting program

```
remove_from_notification_list (domain_id, asset_group_id, user_id)
```

Remove entry from insert completion notification list

This method checks validation only.

#### **Parameters**

- domain\_id (bytes) target domain\_id
- asset\_group\_id (bytes) target asset\_group\_id of which you want to get notification about the insertion
- user id (bytes) user id that registers in the list

Search transactions that match given conditions

When Multiple conditions are given, they are considered as AND condition.

#### **Parameters**

- domain\_id (bytes) target domain\_id
- asset\_group\_id (bytes) asset\_group\_id that target transactions should have
- asset\_id (bytes) asset\_id that target transactions should have
- user\_id (bytes) user\_id that target transactions should have
- count (int) The maximum number of transactions to retrieve

Returns dictionary having transaction\_id, serialized transaction data, asset files

Return type dict

### **Parameters**

- domain id (bytes) target domain id
- asset\_group\_ids (list) list of asset\_group\_ids
- transaction\_id (bytes) transaction\_id that has just inserted
- only registered user (bool) If True, notification is not sent to other nodes

```
validate transaction(txdata, asset files=None)
```

Validate transaction by verifying signature

### **Parameters**

- txdata (bytes) serialized transaction data
- asset\_files (dict) dictionary of {asset\_id: content} for the transaction

**Returns** if validation fails, None returns.

Return type BBcTransaction

```
bbc1.core.bbc_core.activate_ledgersubsystem()
    Load module of ledger_subsystem if installed
bbc1.core.bbc_core.daemonize(pidfile='/tmp/bbc1.pid')
    Run in background
```

## bbc1.core.bbc error module

Copyright (c) 2017 beyond-blockchain.org.

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

```
http://www.apache.org/licenses/LICENSE-2.0
```

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

## bbc1.core.bbc\_network module

Copyright (c) 2017 beyond-blockchain.org.

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

```
http://www.apache.org/licenses/LICENSE-2.0
```

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

```
class bbc1.core.bbc_network.BBcNetwork(config, core=None, p2p_port=None, external_ip4addr=None, external_ip4addr=None, loglevel='all', logname=None)

Bases: object
```

Socket and thread management for infrastructure layers

```
CONFIRM_KEY_EXCHANGE = b'\x00\x03'
```

```
NOTIFY_LEAVE = b'\x00\x00'

REQUEST_KEY_EXCHANGE = b'\x00\x01'

RESPONSE_KEY_EXCHANGE = b'\x00\x02'

add_neighbor (domain_id, node_id, ipv4=None, ipv6=None, port=None, is_static=False)

Add node in the neighbor list
```

#### **Parameters**

- domain\_id (bytes) target domain\_id
- node\_id (bytes) target node\_id
- ipv4 (str) IPv4 address of the node
- ipv6 (str) IPv6 address of the node
- port (int) Port number that the node is waiting at
- is\_static (bool) If true, the entry is treated as static one and will be saved in config.json

**Returns** True if it is a new entry, None if error.

Return type bool

broadcast\_message\_in\_network(domain\_id, payload\_type=1, msg=None)

Send message to all neighbor nodes

#### **Parameters**

- payload\_type (bytes) message format type
- domain\_id (bytes) target domain\_id
- msg (dict) message to send

**Returns** True if successful

Return type bool

## check\_admin\_signature(domain\_id, msg)

Check admin signature in the message

## **Parameters**

- domain\_id (bytes) target domain\_id
- msg (dict) received message

Returns True if valid

**Return type** bool

Create domain and register user in the domain

## **Parameters**

- domain\_id (bytes) target domain\_id to create
- config (dict) confituration for the domain

## Returns

Return type bool

### get\_domain\_keypair (domain\_id)

Get domain\_keys (private key and public key)

Parameters domain\_id (bytes) - target domain\_id

## include\_admin\_info\_into\_message\_if\_needed(domain\_id, msg, admin\_info)

Serialize admin info into one binary object and add signature

**Parameters** domain\_id (bytes) - target domain\_id to remove

Returns True if successful

Return type bool

### save\_all\_static\_node\_list()

Save all static nodes in the config file

## send\_domain\_ping (domain\_id, ipv4, ipv6, port, is\_static=False)

Send domain ping to the specified node

#### **Parameters**

- domain\_id (bytes) target domain\_id
- ipv4 (str) IPv4 address of the node
- ipv6 (str) IPv6 address of the node
- port (int) Port number
- is\_static (bool) If true, the entry is treated as static one and will be saved in config.json

Returns True if successful

Return type bool

Send ECDH key exchange message

**send\_message\_in\_network** (nodeinfo=None, payload\_type=1, domain\_id=None, msg=None) Send message over a domain network

## **Parameters**

- nodeinfo (NodeInfo) NodeInfo object of the destination
- payload\_type (bytes) message format type
- domain\_id (bytes) target domain\_id
- msg (dict) message to send

**Returns** True if successful

Return type bool

## send\_message\_to\_a\_domain0\_manager(domain\_id, msg)

Choose one of domain0\_managers and send msg to it

## **Parameters**

- domain\_id (bytes) target domain\_id
- msg (bytes) message to send

```
setup_tcp_server()
         Start tcp server
     setup_udp_socket()
         Setup UDP socket
     tcpserver_loop()
         Message loop for TCP socket
     udp message loop()
         Message loop for UDP socket
class bbc1.core.bbc_network.NeighborInfo(network=None,
                                                                              domain_id=None,
                                                   node_id=None, my_info=None)
     Bases: object
     Manage information of neighbor nodes
     NODEINFO LIFETIME = 900
     PURGE_INTERVAL_SEC = 300
     add (node_id, ipv4=None, ipv6=None, port=None, is_static=False, domain0=None)
         Add or update an neighbor node entry
     purge (query entry)
         Purge obsoleted entry in nodeinfo_list
     remove (node_id)
         Remove entry in the nodeinfo_list
     show_list()
         Return nodeinfo list in human readable format
class bbc1.core.bbc_network.NodeInfo(node_id=None, ipv4=None, ipv6=None, port=None,
                                              is_static=False, domain0=False)
     Bases: object
     Node information entry
     SECURITY_STATE_CONFIRMING = 2
     SECURITY STATE ESTABLISHED = 3
     SECURITY STATE NONE = 0
     SECURITY_STATE_REQUESTING = 1
     get_nodeinfo()
         Return a list of node info
             Returns [node_id, ipv4, ipv6, port, domain0_flag, update_at]
             Return type list
     touch()
     update (ipv4=None, ipv6=None, port=None, seq=None, domain0=None)
         Update the entry
             Parameters
                 • ipv4 (str) - IPv4 address of the sender node
                 • ipv6 (str) – IPv6 address of the sender node
                 • port (int) – Port number of the sender
```

- **sec** (*int*) message sequence number
- domain 0 (bool or None) If True, the node is domain 0 manager

Returns True if the entry has changed

Return type bool

```
bbc1.core.bbc_network.is_less_than (val_a, val_b)
Return True if val_a is less than val_b (evaluate as integer)
```

## bbc1.core.bbc stats module

Copyright (c) 2017 beyond-blockchain.org.

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

```
http://www.apache.org/licenses/LICENSE-2.0
```

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

```
class bbc1.core.bbc_stats.BBcStats
    Bases: object
    clear_stats()
    get_stats()
    remove_stat_category(category)
    remove_stat_item(category, name)
    update_stats(category, name, value)
    update_stats_decrement(category, name, value)
    update_stats_increment(category, name, value)
```

#### bbc1.core.bbclib module

Copyright (c) 2017 beyond-blockchain.org.

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

```
http://www.apache.org/licenses/LICENSE-2.0
```

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

```
class bbc1.core.bbclib.BBcAsset (user_id=None, asset_file=None, asset_body=None)
    Bases: object
    Asset part in a transaction
    add (user_id=None, asset_file=None, asset_body=None)
    Add parts in this object
```

```
deserialize (data)
          Deserialize into this object
              Parameters data (bytes) – serialized binary data
              Returns True if successful
              Return type bool
     digest()
          Calculate the digest
          The digest corresponds to the asset_id of this object
               Returns asset_id (or digest)
              Return type bytes
     get_asset_file()
          Get asset file content and its digest
              Returns digest of the file content bytes: the file content
              Return type bytes
     recover_asset_file (asset_file)
          Recover asset file info from the given raw content
     serialize (for_digest_calculation=False)
          Serialize this object
class bbc1.core.bbclib.BBcCrossRef (domain id=None,
                                                                    transaction id=None,
                                                                                             deserial-
                                               ize=None)
     Bases: object
     CrossRef part in a transaction
     deserialize (data)
          Deserialize into this object
              Parameters data (bytes) – serialized binary data
              Returns True if successful
              Return type bool
     serialize()
          Serialize this object
class bbc1.core.bbclib.BBcEvent (asset_group_id=None)
     Bases: object
     Event part in a transaction
     add (asset_group_id=None,
                                      reference_index=None,
                                                                  mandatory_approver=None,
                                                                                                  op-
           tion approver num numerator=0,
                                                    option approver num denominator=0,
                                                                                                  op-
           tion approver=None, asset=None)
          Add parts
     deserialize (data)
          Deserialize into this object
              Parameters data (bytes) - serialized binary data
              Returns True if successful
              Return type bool
```

```
serialize()
          Serialize this object
class bbc1.core.bbclib.BBcPointer(transaction_id=None, asset_id=None)
     Bases: object
     Pointer part in a transaction
     add (transaction_id=None, asset_id=None)
          Add parts
     deserialize (data)
          Deserialize into this object
              Parameters data (bytes) – serialized binary data
              Returns True if successful
              Return type bool
     serialize()
          Serialize this object
class bbc1.core.bbclib.BBcReference(asset_group_id, transaction,
                                                                             ref_transaction=None,
                                               event_index_in_ref=0)
     Bases: object
     Reference part in a transaction
     add_signature(user_id=None, signature=None)
          Add signature in the reserved space
              Parameters
                  • user_id (bytes) – user_id of the signature owner
                  • signature (bytes) - signature
     deserialize(data)
          Deserialize into this object
              Parameters data (bytes) – serialized binary data
              Returns True if successful
              Return type bool
     get destinations()
          Return the list of approvers in the referred transaction
     get_referred_transaction()
          Return referred transaction in serialized format
     prepare_reference (ref_transaction)
          Read the previous referencing transaction
     serialize()
          Serialize this object
class bbc1.core.bbclib.BBcRelation(asset_group_id=None)
     Bases: object
     Relation part in a transaction
     add (asset_group_id=None, asset=None, pointer=None)
          Add parts
```

```
deserialize (data)
          Deserialize into this object
              Parameters data (bytes) – serialized binary data
              Returns True if successful
              Return type bool
     serialize()
          Serialize this object
class bbc1.core.bbclib.BBcSignature(key_type=1, deserialize=None)
     Bases: object
     Signature part in a transaction
     add (signature=None, pubkey=None)
          Add signature and public key
     deserialize (data)
          Deserialize into this object
              Parameters data (bytes) – serialized binary data
              Returns True if successful
              Return type bool
     serialize()
          Serialize this object
     verify(digest)
          Verify digest using pubkey in signature
              Parameters digest (bytes) – digest to verify
              Returns 0:invalid, 1:valid
              Return type int
class bbc1.core.bbclib.BBcTransaction(version=0, deserialize=None, jsonload=None)
     Bases: object
     Transaction object
     add (event=None, reference=None, relation=None, witness=None, cross_ref=None)
          Add parts
     add_signature(user_id=None, signature=None)
          Add signature in the reserved space
              Parameters
                  • user_id (bytes) - user_id of the signature owner
                  • signature (bytes) - signature
              Returns True if successful
              Return type bool
     deserialize (data)
          Deserialize into this object
              Parameters data (bytes) – serialized binary data
              Returns True if successful
```

## Return type bool digest() Calculate the digest The digest corresponds to the transaction\_id of this object **Returns** transaction id (or digest) **Return type** bytes get\_sig\_index (user\_id) Reserve a space for signature for the specified user\_id **Parameters user\_id** (bytes) – user\_id whose signature will be added to the signature part **Returns** position (index) in the signature part Return type int jsondump() Dump the transaction in json format jsonload (jsontx) Load the transaction in json format serialize(for\_id=False) Serialize the whole parts sign (key\_type=1, private\_key=None, public\_key=None, keypair=None) Sign the transaction **Parameters** • key\_type (int) - Type of encryption key algorighm (currently, Key-Type.ECDSA\_SECP256k1 only) • private\_key (bytes) -• public\_key (bytes) -• keypair (KeyPair) - keypair or set of private\_key and public\_key needs to be given Returns Return type BBcSignature class bbc1.core.bbclib.BBcWitness Bases: object Witness part in a transaction add\_signature(user\_id=None, signature=None) Add signature in the reserved space for the user\_id that was registered before **Parameters** • user\_id (bytes) – user\_id of the signature owner • signature (bytes) - signature add\_witness(user\_id) Register user\_id in the list deserialize (data) Deserialize into this object

**Parameters data** (bytes) – serialized binary data

```
Returns True if successful
             Return type bool
     serialize()
         Serialize this object
class bbc1.core.bbclib.KeyPair(type=1, privkey=None, pubkey=None)
     Bases: object
     Key pair container
     generate()
         Generate a new key pair
     get_private_key_in_der()
         Return private key in DER format
     get_private_key_in_pem()
         Return private key in PEM format
     mk_keyobj_from_private_key()
         Make a keypair object from the binary data of private key
     mk_keyobj_from_private_key_der(derdat)
         Make a keypair object from the private key in DER format
     mk_keyobj_from_private_key_pem(pemdat_string)
         Make a keypair object from the private key in PEM format
     sign (digest)
         Sign to the given value
             Parameters digest (bytes) - given value
             Returns signature
             Return type bytes
     to_bigint(dat)
     to_binary(dat)
     verify (digest, sig)
         Verify the digest and the signature using the rivate key in this object
class bbc1.core.bbclib.KeyType
     Bases: object
     ECDSA SECP256k1 = 1
class bbc1.core.bbclib.MsgType
     Bases: object
     Message types for between core node and client
     CANCEL_INSERT_NOTIFICATION = 16
     DOMAIN_PING = 12
     MESSAGE = 66
     NOTIFY_CROSS_REF = 74
     NOTIFY_DOMAIN_KEY_UPDATE = 19
     NOTIFY INSERTED = 73
```

```
REGISTER = 64
REQUEST_CLOSE_DOMAIN = 31
REQUEST_CROSS_REF_LIST = 92
REQUEST_CROSS_REF_VERIFY = 90
REQUEST ECDH KEY EXCHANGE = 33
REQUEST GATHER SIGNATURE = 67
REQUEST_GET_CONFIG = 8
REQUEST_GET_DOMAINLIST = 13
REQUEST_GET_FORWARDING_LIST = 25
REQUEST_GET_NEIGHBORLIST = 21
REQUEST\_GET\_NODEID = 27
REQUEST_GET_NOTIFICATION_LIST = 29
REQUEST\_GET\_STATS = 17
REQUEST\_GET\_USERS = 23
REQUEST_INSERT = 71
REQUEST INSERT NOTIFICATION = 15
REQUEST MANIP LEDGER SUBSYS = 10
REQUEST_REGISTER_HASH_IN_SUBSYS = 128
REQUEST_REPAIR = 94
REQUEST SEARCH TRANSACTION = 82
REQUEST_SEARCH_WITH_CONDITIONS = 86
REQUEST_SETUP_DOMAIN = 0
REQUEST_SET_STATIC_NODE = 4
REQUEST SIGNATURE = 69
REQUEST TRAVERSE TRANSACTIONS = 88
REQUEST_VERIFY_HASH_IN_SUBSYS = 130
RESPONSE_CLOSE_DOMAIN = 32
RESPONSE_CROSS_REF_LIST = 93
RESPONSE_CROSS_REF_VERIFY = 91
RESPONSE_ECDH_KEY_EXCHANGE = 34
RESPONSE_GATHER_SIGNATURE = 68
RESPONSE_GET_CONFIG = 9
RESPONSE_GET_DOMAINLIST = 14
RESPONSE_GET_FORWARDING_LIST = 26
RESPONSE GET NEIGHBORLIST = 22
```

RESPONSE\_GET\_NODEID = 28

```
RESPONSE GET NOTIFICATION LIST = 30
    RESPONSE GET STATS = 18
    RESPONSE_GET_USERS = 24
    RESPONSE INSERT = 72
    RESPONSE MANIP LEDGER SUBSYS = 11
    RESPONSE REGISTER HASH IN SUBSYS = 129
    RESPONSE_SEARCH_TRANSACTION = 83
    RESPONSE_SEARCH_WITH_CONDITIONS = 87
    RESPONSE_SETUP_DOMAIN = 1
    RESPONSE_SET_STATIC_NODE = 5
    RESPONSE_SIGNATURE = 70
    RESPONSE_TRAVERSE_TRANSACTIONS = 89
    RESPONSE VERIFY HASH IN SUBSYS = 131
    UNREGISTER = 65
bbc1.core.bbclib.add_event_asset(transaction, event_idx, asset_group_id, user_id,
                                       set_body=None, asset_file=None)
    Utility to add BBcEvent object with BBcAsset in the transaction
bbc1.core.bbclib.add reference to transaction (transaction,
                                                                            asset group id,
                                                       ref_transaction_obj, event_index_in_ref)
    Utility to add BBcReference object in the transaction
         Returns
         Return type BBcReference
bbc1.core.bbclib.add_relation_asset(transaction, relation_idx, asset_group_id, user_id, as-
                                           set_body=None, asset_file=None)
    Utility to add BBcRelation object with BBcAsset in the transaction
bbc1.core.bbclib.add_relation_pointer(transaction, relation_idx, ref_transaction_id=None,
                                             ref asset id=None)
    Utility to add BBcRelation object with BBcPointer in the transaction
bbc1.core.bbclib.bin2str_base64(dat)
bbc1.core.bbclib.convert_id_to_string(data, bytelen=32)
    Convert binary data to hex string
bbc1.core.bbclib.convert_idstring_to_bytes (datastr, bytelen=32)
    Convert hex string to binary data
bbc1.core.bbclib.get_bigint(ptr, dat)
bbc1.core.bbclib.get_n_byte_int(ptr, n, dat)
bbc1.core.bbclib.get_n_bytes(ptr, n, dat)
bbc1.core.bbclib.get_new_id(seed_str=None, include_timestamp=True)
    Return 256-bit binary data
         Parameters
              • seed_str (str) – seed string that is hashed by SHA256
```

• include\_timestamp (bool) - if True, timestamp (current time) is appended to the seed string

**Returns** 256-bit binary

Return type bytes

bbc1.core.bbclib.get\_random\_id()

Return 256-bit binary data

**Returns** 256-bit random binary

Return type bytes

bbc1.core.bbclib.get\_random\_value(length=8)

Return 1-byte random value

bbc1.core.bbclib.make\_transaction(event\_num=0, relation\_num=0, witness=False)
Utility to make transaction object

### **Parameters**

- event\_num (int) the number of BBcEvent object to include in the transaction
- relation\_num (int) the number of BBcRelation object to include in the transaction
- witness (bool) If true, BBcWitness object is included in the transaction

#### Returns

Return type BBcTransaction

#### **Parameters**

- txobj (BBcTransaction) target transaction object
- **asset\_files** (dict) dictionary containing the asset file contents

**Returns** True if valid tuple: list of valid assets tuple: list of invalid assets

Return type bool

```
bbc1.core.bbclib.verify_using_cross_ref(domain_id, transaction_id, transaction_base_digest, cross_ref_data, sigdata)

Confirm the existence of the transaction using cross_ref
```

#### **Parameters**

- domain\_id (bytes) target domain\_id
- transaction\_id (bytes) target transaction\_id of which existence you want to confirm
- transaction\_base\_digest (bytes) digest obtained from the outer domain
- cross\_ref\_data (bytes) serialized BBcCrossRef object
- **sigdata** (bytes) serialized signature

Returns True if valid

Return type bool

### bbc1.core.command module

Copyright (c) 2017 beyond-blockchain.org.

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

```
http://www.apache.org/licenses/LICENSE-2.0
```

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

```
bbc1.core.command.parser()
```

## bbc1.core.data\_handler module

Copyright (c) 2017 beyond-blockchain.org.

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

```
http://www.apache.org/licenses/LICENSE-2.0
```

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

## **Parameters**

- **db\_num** (int) index of DB if multiple DBs are used
- **sql** (*str*) **SQL** string
- args (list) Args for the SQL
- commit (bool) If True, commit is performed
- **fetch\_one** (bool) If True, fetch just one record

**Returns** list of records

Return type list

## get\_in\_storage (asset\_group\_id, asset\_id)

Get the asset file with the asset\_id from local storage

## **Parameters**

- asset\_group\_id (bytes) asset\_group\_id of the asset
- asset\_id (bytes) asset\_id of the asset

Returns the file content

Return type bytes or None

#### **Parameters**

- transaction\_id (bytes) target transaction\_id
- outer\_domain\_id (bytes) domain\_id that holds cross\_ref about the transaction\_id
- txid\_having\_cross\_ref (bytes) transaction\_id in the outer\_domain that includes the cross\_ref
- no\_replication (bool) If False, the replication is sent to other nodes in the domain

insert\_transaction (txdata, txobj=None, asset\_files=None, no\_replication=False)

Insert transaction data and its asset files

Either txdata or txobj must be given to insert the transaction.

## **Parameters**

• **txdata** (bytes) – serialized transaction data

- txobj (BBcTransaction) transaction object to insert
- asset\_files (dict) asset files in the transaction

Returns set of asset\_group\_ids in the transaction

Return type set

#### process message (msg)

Process received message

**Parameters** msg (dict) – received message

remove (transaction\_id, txobj=None, db\_num=-1)

Delete all data regarding the specified transaction\_id

This method requires either transaction\_id or txobj.

### **Parameters**

- transaction\_id (bytes) target transaction\_id
- txobj (BBcTransaction) transaction object to remove
- **db\_num** (*int*) index of DB if multiple DBs are used

## restore\_transaction\_data(db\_num, transaction\_id, txobj)

Remove and insert a transaction

#### search\_domain\_having\_cross\_ref(transaction\_id=None)

Search domain\_id that holds cross\_ref about the specified transaction\_id

Parameters transaction\_id (bytes) - target transaction\_id

**Returns** records of cross\_ref\_tables ["id","transaction\_id", "outer\_domain\_id", "txid\_having\_cross\_ref"]

Return type list

Search transaction data

When Multiple conditions are given, they are considered as AND condition.

## **Parameters**

- transaction\_id (bytes) target transaction\_id
- asset\_group\_id (bytes) asset\_group\_id that target transactions should have
- asset\_id (bytes) asset\_id that target transactions should have
- user\_id (bytes) user\_id that target transactions should have
- **count** (*int*) The maximum number of transactions to retrieve
- **db\_num** (*int*) index of DB if multiple DBs are used

**Returns** mapping from transaction\_id to serialized transaction data dict: dictionary of {asset\_id: content} for the transaction

Return type dict

## search\_transaction\_topology (transaction\_id, traverse\_to\_past=True)

Search in topology info

## **Parameters**

- transaction\_id (bytes) base transaction\_id
- **traverse\_to\_past** (bool) True: search backward (to past), False: search forward (to future)

**Returns** list of records of topology table

**Return type** list

 $\verb|store_in_storage| (asset\_group\_id, asset\_id, content, do\_overwrite = False)|$ 

Store asset file in local storage

#### **Parameters**

- asset\_group\_id (bytes) asset\_group\_id of the asset
- asset\_id (bytes) asset\_id of the asset
- content (bytes) the content of the asset file
- do\_overwrite (bool) If True, file is overwritten

Returns True if successful

Return type bool

Bases: bbc1.core.data\_handler.DataHandler

Data handler for domain\_global\_0

exec\_sql (sql, \*args)

Execute sql sentence

## **Parameters**

- **db\_num** (*int*) index of DB if multiple DBs are used
- sql (str) SQL string
- args (list) Args for the SQL
- commit (bool) If True, commit is performed
- **fetch\_one** (bool) If True, fetch just one record

**Returns** list of records

Return type list

get\_in\_storage (asset\_group\_id, asset\_id)

Get the asset file with the asset\_id from local storage

## **Parameters**

- $asset\_group\_id$  (bytes)  $asset\_group\_id$  of the asset
- $asset_id(bytes)$   $asset_id$  of the asset

**Returns** the file content

**Return type** bytes or None

insert\_transaction (txdata, txobj=None, asset\_files=None, no\_replication=False)

Insert transaction data and its asset files

Either txdata or txobj must be given to insert the transaction.

## **Parameters**

- txdata (bytes) serialized transaction data
- txobj (BBcTransaction) transaction object to insert
- asset\_files (dict) asset files in the transaction

**Returns** set of asset\_group\_ids in the transaction

#### **Return type** set

## process\_message (msg)

Process received message

**Parameters** msg (dict) – received message

remove (transaction\_id)

Delete all data regarding the specified transaction\_id

This method requires either transaction\_id or txobj.

#### **Parameters**

- transaction\_id (bytes) target transaction\_id
- txobj (BBcTransaction) transaction object to remove
- **db\_num** (*int*) index of DB if multiple DBs are used

Search transaction data

When Multiple conditions are given, they are considered as AND condition.

## **Parameters**

- transaction\_id (bytes) target transaction\_id
- $asset\_group\_id$  (bytes)  $asset\_group\_id$  that target transactions should have
- asset\_id (bytes) asset\_id that target transactions should have
- user\_id (bytes) user\_id that target transactions should have
- count (int) The maximum number of transactions to retrieve
- **db\_num** (*int*) index of DB if multiple DBs are used

**Returns** mapping from transaction\_id to serialized transaction data dict: dictionary of {asset\_id: content} for the transaction

## Return type dict

```
search_transaction_topology (transaction_id, reverse_link=False)
```

Search in topology info

## **Parameters**

- **transaction\_id** (bytes) base transaction\_id
- **traverse\_to\_past** (bool) True: search backward (to past), False: search forward (to future)

**Returns** list of records of topology table

Return type list

```
store_in_storage (asset_group_id, asset_id, content)
          Store asset file in local storage
              Parameters
                  • asset_group_id (bytes) - asset_group_id of the asset
                  • asset id (bytes) - asset id of the asset
                  • content (bytes) – the content of the asset file
                  • do_overwrite (bool) - If True, file is overwritten
              Returns True if successful
              Return type bool
class bbc1.core.data_handler.DbAdaptor(handler=None,
                                                                  db_name=None,
                                                                                     db_num=0,
                                                  loglevel='all', logname=None)
     Bases: object
     Base class for DB adaptor
     check_table_existence(tblname)
          Check whether the table exists or not
     create table (tbl, tbl definition, primary key=0, indices=[])
          Create a table
     open db()
          Open the DB
class bbc1.core.data_handler.MysqlAdaptor(handler=None,
                                                                                db_name=None,
                                                      db num=None,
                                                                               server_info=None,
                                                      loglevel='all', logname=None)
     Bases: bbc1.core.data handler.DbAdaptor
     DB adaptor for MySQL
     check_table_existence(tblname)
          Check whether the table exists or not
     create table (tbl, tbl definition, primary key=0, indices=[])
          Create a table
              Parameters
                  • tbl (str) - table name
                  • tbl definition (list) - schema of the table [["column name", "data
                   type"],["colmun_name", "data type"],,]
                  • primary_key (int) - index (column) of the primary key of the table
                  • indices (list) – list of indices to create index
     open db()
          Open the DB
class bbc1.core.data_handler.SqliteAdaptor(handler=None,
                                                                                db name=None,
                                                       loglevel='all', logname=None)
     Bases: bbc1.core.data_handler.DbAdaptor
     DB adaptor for SQLite3
     check_table_existence(tblname)
          Check whether the table exists or not
```

```
create_table (tbl, tbl_definition, primary_key=0, indices=[])
Create a table
```

#### **Parameters**

- **tbl** (str) table name
- **tbl\_definition** (*list*) schema of the table [["column\_name", "data type"],["colmun\_name", "data type"],]
- primary\_key (int) index (column) of the primary key of the table
- indices (list) list of indices to create index

open\_db()

Open the DB (create DB file if not exists)

#### bbc1.core.domain0 manager module

Copyright (c) 2017 beyond-blockchain.org.

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

```
http://www.apache.org/licenses/LICENSE-2.0
```

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

```
class bbc1.core.domain0_manager.Domain0Manager(networking=None,
                                                                          node_id=None,
                                                       loglevel='all', logname=None)
    Bases: object
    Management for inter-domain collaboration over domain_global_0
    ADV DOMAIN LIST = b' \times 00 \times 00'
    CROSS REF PROBABILITY = 0.1
    DISTRIBUTE CROSS REF = b'\x00\x01'
    DOMAIN_ACCEPTANCE_RECOVER_INTERVAL = 600
    DOMAIN INFO ADVERTISE INTERVAL = 1800
    DOMAIN_INFO_LIFETIME = 3600
    INITIAL_ACCEPT_LIMIT = 10
    NOTIFY_CROSS_REF_REGISTERED = b'\x00\x02'
    NUM_OF_COPIES = 3
    REQUEST_VERIFY = b' \times 00 \times 04'
    REQUEST_VERIFY_FROM_OUTER_DOMAIN = b'\x00\x05'
    RESPONSE VERIFY FROM OUTER DOMAIN = b'\x00\x06'
    cross ref registered (domain id, transaction id, cross ref)
         Notify cross_ref inclusion in a transaction of the outer domain and insert the info into DB
```

## **Parameters**

• domain\_id (bytes) - domain\_id where the cross\_ref is from

```
• transaction_id (bytes) - transaction_id that the cross_ref proves
```

• cross ref (bytes) - the registered cross ref in other domain

## distribute\_cross\_ref\_in\_domain0 (domain\_id, transaction\_id)

Determine if the node distributes the cross ref (into domain global 0)

#### **Parameters**

- domain\_id (bytes) target domain\_id
- transaction\_id (bytes) target transaction\_id

#### process\_message (msg)

Process received message

Parameters msg (dict) - received message

```
stop_all_timers()
```

Invalidate all running timers

#### update\_domain\_belong\_to()

Update the list domain\_belong\_to

domain\_belong\_to holds all domain\_ids that this node belongs to

#### bbc1.core.key exchange manager module

Copyright (c) 2017 beyond-blockchain.org.

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

```
http://www.apache.org/licenses/LICENSE-2.0
```

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

```
class bbc1.core.key_exchange_manager.KeyExchangeManager (networking,
                                                                            domain_id,
                                                                counter node id)
    Bases: object
    ECDH (Elliptic Curve Diffie-Hellman) key exchange manager
    KEY EXCHANGE INVOKE MAX BACKOFF = 6
    KEY_EXCHANGE_RETRY_INTERVAL = 5
    KEY_OBSOLETE_TIMER = 10
    KEY_REFRESH_INTERVAL = 604800
    STATE CONFIRMING = 2
    STATE\_ESTABLISHED = 3
    STATE_NONE = 0
    STATE REQUESTING = 1
    receive confirmation()
         Confirm that the key has been agreed
    receive_exchange_request (pubkey, nonce, random_val, hint)
```

Procedure when receiving message with BBcNetwork.REQUEST\_KEY\_EXCHANGE

#### **Parameters**

```
    pubkey (bytes) - public key
    nonce (bytes) - nonce value
    random_val (bytes) - random value in calculating key
    receive_exchange_response (pubkey, random_val, hint)
    Process ECDH procedure (receiving response)
    set_cipher (key_name, hint)
    Set key to the encryptor and decryptor
    set_invoke_timer (timeout, retry_entry=False)
    Set timer for key refreshment
    stop_all_timers ()
    Stop all timers
    unset_cipher (key_name=None)
    Unset key from the encryptor and decryptor
    bbc1.core.key_exchange_manager.remove_old_key (query_entry)
```

## bbc1.core.logger module

Copyright (c) 2017 beyond-blockchain.org.

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

```
http://www.apache.org/licenses/LICENSE-2.0
```

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

```
bbc1.core.logger.get_logger(key=", logname='-', level='none')
```

#### bbc1.core.message\_key\_types module

Copyright (c) 2017 beyond-blockchain.org.

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

```
http://www.apache.org/licenses/LICENSE-2.0
```

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

```
class bbc1.core.message_key_types.InfraMessageCategory
    Bases: object
    Types of message for inter-core nodes messaging
    CATEGORY_DATA = b'\x00\x03'
    CATEGORY_DOMAINO = b'\x00\x04'
```

```
CATEGORY NETWORK = b' \times 00 \times 00'
     CATEGORY TOPOLOGY = b' \times 00 \times 01'
     CATEGORY_USER = b' \times 00 \times 02'
class bbc1.core.message_key_types.KeyType
     Bases: object
     Types of items in a message
     admin_info = b'\x00\x00\x00\x17'
     all_asset_files = b'\x00\x00\x00u'
     all included = b' \times 00 \times 00 \times 00h'
     anycast_ttl = b'\x00\x00\x00\x1a'
     asset_file = b'\x00\x00\x00t'
     asset\_group\_id = b'\x00\x00\x00c'
     asset group ids = b' \times 00 \times 00 \times 00d'
     asset id = b'\x00\x00\x00e'
     bbc\_configuration = b'\x00\x00\x'
     command = b' \times 00 \times 00 \times 10^{-1}
     compromised_asset_files = b'\x00\x00\x00\x92'
     compromised_transaction_data = b'\x00\x00\x00\x90'
     compromised_transactions = b'\x00\x00\x00\x91'
     count = b' \times 00 \times 00 \times 00 \times 0e'
     cross\_ref = b' \x00 \x00 \x00w'
     cross_ref_verification_info = b'\x00\x00\x00{'
     destination\_node\_id = b'\x00\x00\x00V'
     destination user id = b' \times 00 \times 00 \times 00R'
     destination_user_ids = b'\x00\x00\x00S'
     direction = b' \times 00 \times 00 \times 00f'
     domain_id = b' \times 00 \times 00 \times 00P'
     domain list = b' \times 00 \times 00 \times 007'
     domain ping = b' \times 00 \times 00 \times 15'
     ecdh = b' \times 00 \times 00 \times 11'
     external_ip4addr = b'\x00\x00\x004'
     external_ip6addr = b'\x00\x00\x005'
     forwarding_list = b' \times 00 \times 00 \times 008'
     hint = b' \times 00 \times 00 \times 10'
     hop\_count = b' \times 00 \times 00 \times 00g'
     infra\_command = b'\x00\x00\x00\n'
```

```
infra_msg_type = b'\x00\x00\x00\x00
ipv4\_address = b'\x00\x00\x001'
ipv6\_address = b'\x00\x00\x002'
is\_anycast = b'\x00\x00\x00\x19'
is replication = b' \times 00 \times 00 \times 1b'
ledger subsys manip = b' \times 00 \times 00 \times 00 \times 00
ledger_subsys_register = b'\x00\x00\x00\xa1'
ledger_subsys_verify = b'\x00\x00\x00\xa2'
merkle\_tree = b'\x00\x00\x00\xa3'
message = b'\x00\x00\x00\x00
message_seq = b'\x00\x00\x00\x14'
neighbor_list = b' \times 00 \times 00 \times 00:
node_id = b' \times 00 \times 00 \times 00T'
node info = b'\x00\x00\x006'
nodekey_signature = b'\x00\x00\x16'
nonce = b'\x00\x00\x00\r'
notification_list = b'\x00\x00\x00;'
on_multinodes = b'\x00\x00\x00\x18'
outer_domain_id = b' \times 00 \times 00 \times 00
port_number = b' \times 00 \times 00 \times 003'
query_id = b'\x00\x00\x00\x0b'
random = b' \times 00 \times 00 \times 12'
reason = b' \times 00 \times 00 \times 01'
ref index = b'\x00\x00\x00s'
result = b'\x00\x00\x00\x02'
retry_timer = b'\x00\x00\x00\x13'
signature = b' \times 00 \times 00 \times 00v'
source\_domain\_id = b'\x00\x00\x00y'
source node id = b' \times 00 \times 00 \times 00U'
source_user_id = b'\x00\x00\x000'
static\_entry = b' \times 00 \times 000 \times 000'
stats = b' \times 00 \times 00 \times 00
status = b' \times 00 \times 00 \times 00'
transaction_data = b'\x00\x00\x00p'
transaction id = b'\x00\x00\x00a'
transaction id list = b'\x00\x00\x00b'
```

```
transaction_tree = b' \times 00 \times 00 \times 00r'
     transactions = b' \times 00 \times 00 \times 00q'
     txid_having_cross_ref = b'\x00\x00\x00z'
     user_id = b'\x00\x00\x00'
     user list = b' \times 00 \times 00 \times 009'
class bbc1.core.message_key_types.Message
     Bases: object
     Message parser
     HEADER LEN = 8
     parse()
         Parse the message in the buffer
     recv(dat)
         Append message to the buffer
class bbc1.core.message_key_types.PayloadType
     Bases: object
     Type_any = 1
     Type\_binary = 0
     Type_encrypted_msgpack = 3
     Type_msgpack = 2
bbc1.core.message_key_types.convert_from_binary(data_type, dat)
     Deserialization from simple serialization
bbc1.core.message_key_types.derive_shared_key(private_key,
                                                                            serialized_pubkey,
                                                         shared_info)
     Utility for deriving shared key in ECDH procedure
bbc1.core.message_key_types.deserialize_data(payload_type, dat)
     Utility for deserializing the received message
bbc1.core.message_key_types.get_ECDH_parameters()
     Utility for initialization of ECDH parameters
bbc1.core.message_key_types.make_TLV_formatted_message(msg)
     Utility for simple serialization function
bbc1.core.message_key_types.make_binary(dat)
     Simple serialize function
     Basically, Type-Length-Value format is created for each item.
bbc1.core.message_key_types.make_dictionary_from_TLV_format(dat)
     Utility for simple deserialization function
bbc1.core.message_key_types.make_message(payload_type,
                                                                   msg,
                                                                           payload_version=0,
                                                  key_name=None)
     Utility for making serialized message data
bbc1.core.message_key_types.set_cipher(shared_key, nonce, key_name, hint)
     Set shared key to the encryptor and decryptor
     Encryptor and Decryptor are created for each inter-node connection
```

```
bbc1.core.message_key_types.to_2byte(val, offset=0)
bbc1.core.message_key_types.to_4byte(val, offset=0)
bbc1.core.message_key_types.unset_cipher(key_name)
```

#### bbc1.core.query\_management module

Copyright (c) 2017 beyond-blockchain.org.

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

```
http://www.apache.org/licenses/LICENSE-2.0
```

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

```
class bbc1.core.query_management.QueryEntry(expire_after=30,
                                                                               callback expire=None,
                                                           callback=None.
                                                                               callback error=None,
                                                           interval=0, data=\{\}, retry count=-1\}
     Bases: object
     Callback manager
     callback()
          Call a callback function for successful case
     callback_error()
          Call a callback function for failure case
     deactivate()
          Deactivate the entry
     update (fire_after=None, expire_after=None, callback=None, callback_error=None, init=False)
          Update the entry information
              Parameters
                   • fire_after (float) - set callback (periodical) to fire after given time (in second)
                   • expire_after (float) – set expiration timer to given time (in second)
                   • callback (ob i) – callback method that will be called periodically
                   • callback_error (ob j) - callback method that will be called when error happens
                   • init (bool) - If True, the scheduler is sorted again
     update_expiration_time (expire_after)
          Update the expire timer
              Parameters expire_after (float) – new expiration time in second
class bbc1.core.query_management.Ticker(tick_interval=0.049)
     Bases: object
     Clock ticker for query timers
     del_entry(nonce)
          Delete an entry from the scheduler identified by nonce
     get_entry(nonce)
          Get an entry identified by nonce
```

```
bbc1.core.query_management.get_ticker(tick_interval=0.049)
```

#### bbc1.core.repair manager module

Copyright (c) 2017 beyond-blockchain.org.

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

```
http://www.apache.org/licenses/LICENSE-2.0
```

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

## bbc1.core.topology manager module

Copyright (c) 2017 beyond-blockchain.org.

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

```
http://www.apache.org/licenses/LICENSE-2.0
```

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

Network topology management for a domain

This class defines how to create topology, meaning that who should be neighbors and provides very simple topology management, that is full mesh topology. If P2P routing algorithm is needed, you should override this class to upgrade functions. This class does not manage the neighbor list itself (It's in BBcNetwork)

```
NEIGHBOR LIST REFRESH INTERVAL = 300
NOTIFY NEIGHBOR LIST = b' \times 00 \times 00'
make_neighbor_list()
    make nodelist binary for advertising
notify_neighbor_update (node_id, is_new=True)
    Update expiration timer for the notified node_id
        Parameters
            • node_id (bytes) - target node_id
            • is_new (bool) – If True, this node is a new comer node
process_message (msg)
    Process received message
        Parameters msg (dict) – received message
stop_all_timers()
    Invalidate all running timers
update refresh timer entry (new entry=True, force refresh time=None)
    Update expiration timer
```

#### bbc1.core.user message routing module

Copyright (c) 2017 beyond-blockchain.org.

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

```
http://www.apache.org/licenses/LICENSE-2.0
```

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

```
class bbc1.core.user_message_routing.UserMessageRouting (networking, loglevel='all', loglevel='all', name=None)

Bases: object

Handle message for clients

CROSS_REF_ASSIGNMENT = b'\x00\x05'

JOIN_MULTICAST_RECEIVER = b'\x00\x03'

LEAVE_MULTICAST_RECEIVER = b'\x00\x04'

MAX_CROSS_REF_STOCK = 10

REFRESH_FORWARDING_LIST_INTERVAL = 300

RESOLVE_TIMEOUT = 5

RESOLVE_USER_LOCATION = b'\x00\x00'
```

```
RESPONSE NO SUCH USER = b' \times 00 \times 02'
     RESPONSE USER LOCATION = b' \times 00 \times 01'
     process_message (msg)
          Process received message
              Parameters msq (dict) - received message
     register_user (user_id, socket, on_multiple_nodes=False)
          Register user to forward message
              Parameters
                  • user_id (bytes) - user_id of the client
                  • socket (Socket) – socket for the client
                  • on_multiple_nodes (bool) – If True, the user_id is also registered in other nodes,
                   meaning multicasting.
     send_message_to_user (msg, direct_only=False)
          Forward message to connecting user
              Parameters
                  • msg (dict) - message to send
                  • direct_only (bool) – If True, _forward_message_to_another_node is not called.
     send_multicast_join (user_id, permanent=False)
          Broadcast JOIN MULTICAST RECEIVER
     send_multicast_leave (user_id)
          Broadcast LEAVE_MULTICAST_RECEIVER
     set aes name(socket, name)
          Set name for specifying AES key for message encryption
              Parameters
                  • socket (Socket) - socket for the client
                  • name (bytes) - name of the client (4-byte random value generated in mes-
                   sage_key_types.get_ECDH_parameters)
     stop_all_timers()
          Cancel all running timers
     unregister_user (user_id, socket)
          Unregister user from the list and delete AES key if exists
              Parameters
                  • user_id (bytes) - user_id of the client
                  • socket (Socket) - socket for the client
class bbc1.core.user_message_routing.UserMessageRoutingDummy (networking,
                                                                               domain_id,
                                                                               loglevel='all',
                                                                               logname=None)
     Bases: bbc1.core.user_message_routing.UserMessageRouting
     Dummy class for bbc_core.py
```

#### process\_message (msg)

Process received message

**Parameters** msg (dict) – received message

register\_user (user\_id, socket, on\_multiple\_nodes=False)

Register user to forward message

#### **Parameters**

- user\_id (bytes) user\_id of the client
- socket (Socket) socket for the client
- on\_multiple\_nodes (bool) If True, the user\_id is also registered in other nodes, meaning multicasting.

### send\_message\_to\_user (msg, direct\_only=False)

Forward message to connecting user

#### **Parameters**

- msg (dict) message to send
- **direct\_only** (bool) If True, \_forward\_message\_to\_another\_node is not called.

#### send\_multicast\_join (user\_id, permanent=False)

Broadcast JOIN\_MULTICAST\_RECEIVER

#### stop\_all\_timers()

Cancel all running timers

### unregister\_user (user\_id, socket=None)

Unregister user from the list and delete AES key if exists

## **Parameters**

- user\_id (bytes) user\_id of the client
- **socket** (Socket) socket for the client

bbc1.core.user\_message\_routing.direct\_send\_to\_user(sock, msg, name=None)

#### 1.1.1.3 Module contents

## 1.2 Module contents

1.2. Module contents 43

# CHAPTER 2

## Indices and tables

- genindex
- modindex
- search

## Python Module Index

## b

```
bbc1,43
bbc1.core,43
bbc1.core.bbc_app, 2
bbc1.core.bbc_config, 12
bbc1.core.bbc_core, 12
bbc1.core.bbc_error,14
bbc1.core.bbc_network, 14
bbc1.core.bbc_stats, 18
bbc1.core.bbclib, 18
bbc1.core.command, 27
bbc1.core.data_handler,27
bbc1.core.domain0_manager,33
bbc1.core.key_exchange_manager, 34
bbc1.core.libbbcsig, 2
bbc1.core.libbbcsig.pybbcsig,1
bbc1.core.logger,35
bbc1.core.message_key_types,35
bbc1.core.query_management, 39
bbc1.core.repair_manager,40
bbc1.core.topology_manager,40
bbc1.core.user_message_routing,41
```

48 Python Module Index

A	attribute), 36
activate_ledgersubsystem() (in module	asset_id (bbc1.core.message_key_types.KeyType attribute), 36
bbc1.core.bbc_core), 14 add() (bbc1.core.bbc_network.NeighborInfo method), 17	
add() (bbc1.core.bbclib.BBcAsset method), 18	В
add() (bbc1.core.bbclib.BBcEvent method), 19	bbc1 (module), 43
add() (bbc1.core.bbclib.BBcPointer method), 20	bbc1.core (module), 43
add() (bbc1.core.bbclib.BBcRelation method), 20	bbc1.core.bbc_app (module), 2
add() (bbc1.core.bbclib.BBcSignature method), 21	bbc1.core.bbc_config (module), 12
add() (bbc1.core.bbclib.BBcTransaction method), 21	bbc1.core.bbc_core (module), 12
add_event_asset() (in module bbc1.core.bbclib), 25	bbc1.core.bbc_error (module), 14
add_neighbor() (bbc1.core.bbc_network.BBcNetwork	bbc1.core.bbc_network (module), 14
method), 15	bbc1.core.bbc_stats (module), 18
add_reference_to_transaction() (in module	bbc1.core.bbclib (module), 18
bbc1.core.bbclib), 25	bbc1.core.command (module), 27
add_relation_asset() (in module bbc1.core.bbclib), 25	bbc1.core.data_handler (module), 27
add_relation_pointer() (in module bbc1.core.bbclib), 25	bbc1.core.domain0_manager (module), 33
add_signature() (bbc1.core.bbclib.BBcReference	bbc1.core.key_exchange_manager (module), 34
method), 20	bbc1.core.libbbcsig (module), 2
add_signature() (bbc1.core.bbclib.BBcTransaction	bbc1.core.libbbcsig.pybbcsig (module), 1
method), 21	bbc1.core.logger (module), 35
add_signature() (bbc1.core.bbclib.BBcWitness method),	bbc1.core.message_key_types (module), 35
22	bbc1.core.query_management (module), 39
add_witness() (bbc1.core.bbclib.BBcWitness method), 22	bbc1.core.repair_manager (module), 40
admin_info (bbc1.core.message_key_types.KeyType at-	bbc1.core.topology_manager (module), 40
tribute), 36	bbc1.core.user_message_routing (module), 41
ADV_DOMAIN_LIST (bbc1.core.domain0_manager.Dom attribute), 33	abhomonfiguration (bbc1.core.message_key_types.KeyType attribute), 36
all_asset_files (bbc1.core.message_key_types.KeyType	BBcAppClient (class in bbc1.core.bbc_app), 2
attribute), 36	BBcAsset (class in bbc1.core.bbclib), 18
all_included (bbc1.core.message_key_types.KeyType at-	BBcConfig (class in bbc1.core.bbc_config), 12
tribute), 36	BBcCoreService (class in bbc1.core.bbc_core), 12
anycast_ttl (bbc1.core.message_key_types.KeyType at-	BBcCrossRef (class in bbc1.core.bbclib), 19
tribute), 36	BBcEvent (class in bbc1.core.bbclib), 19
asset_file (bbc1.core.message_key_types.KeyType	BBcNetwork (class in bbc1.core.bbc_network), 14
attribute), 36	BBcPointer (class in bbc1.core.bbclib), 20
asset_group_id (bbc1.core.message_key_types.KeyType	BBcReference (class in bbc1.core.bbclib), 20
attribute), 36	BBcRelation (class in bbc1.core.bbclib), 20
asset_group_ids (bbc1.core.message_key_types.KeyType	BBcSignature (class in bbc1.core.bbclib), 21
	BBcStats (class in bbc1.core.bbc_stats), 18

tribute), 14

BBcTransaction (class in bbc1.core.bbclib), 21	convert_from_binary() (in module
BBcWitness (class in bbc1.core.bbclib), 22 bin2str_base64() (in module bbc1.core.bbclib), 25	bbc1.core.message_key_types), 38 convert_from_der() (bbc1.core.libbbcsig.pybbcsig.PyBBcSigSecp256k1
broadcast_message_in_network()	method), 1
(bbc1.core.bbc_network.BBcNetwork	convert_from_pem() (bbc1.core.libbbcsig.pybbcsig.PyBBcSigSecp256k1
method), 15	method), 1
0	convert_id_to_string() (in module bbc1.core.bbclib), 25
C	convert_idstring_to_bytes() (in module bbc1.core.bbclib),
Callback (class in bbc1.core.bbc_app), 8	25
callback() (bbc1.core.query_management.QueryEntry method), 39	count (bbc1.core.message_key_types.KeyType attribute),  36
callback_error() (bbc1.core.query_management.QueryEntr	ycount_domain_in_cross_ref()
method), 39	(bbc1.core.data_handler.DataHandler method),
cancel_insert_completion_notification()	create_domain() (bbc1.core.bbc_network.BBcNetwork
(bbc1.core.bbc_app.BBcAppClient method), 2	method), 15
CANCEL_INSERT_NOTIFICATION (bbc1.core.bbclib.MsgType attribute), 23	create_queue() (bbc1.core.bbc_app.Callback method), 8
CATEGORY_DATA (bbc1.core.message_key_types.InfraN	
attribute), 35	method), 32
CATEGORY_DOMAIN0	create_table() (bbc1.core.data_handler.MysqlAdaptor
(bbc1.core.message_key_types.InfraMessageCate	egory method), 32
attribute), 35	create_table() (bbc1.core.data_handler.SqliteAdaptor
CATEGORY_NETWORK	method), 32
(bbc1.core.message_key_types.InfraMessageCate	gpps_ref (bbc1.core.message_key_types.KeyType
attribute), 35	attribute), 36
CATEGORY_TOPOLOGY	CROSS_REF_ASSIGNMENT
(bbc1.core.message_key_types.InfraMessageCate	
attribute), 36	attribute), 41
CATEGORY_USER (bbc1.core.message_key_types.InfraN	(bbc1.core.domain0_manager.Domain0Manager
attribute), 36	attribute), 33
check_admin_signature() (bbc1.core.bbc_network.BBcNetwork	cross_ref_registered() (bbc1.core.domain0_manager.Domain0Manager
method), 15	method), 33
check_table_existence() (bbc1.core.data_handler.DbAdapto	
method), 32	(bbc1.core.message_key_types.KeyType
check_table_existence() (bbc1.core.data_handler.MysqlAda	aptor attribute), 36
method), 32	n
check_table_existence() (bbc1.core.data_handler.SqliteAda	p <del>to</del> r
method), 32	daemonize() (in module bbc1.core.bbc_core), 14
clear_stats() (bbc1.core.bbc_stats.BBcStats method), 18	DataHandler (class in bbc1.core.data_handler), 27
command (bbc1.core.message_key_types.KeyType at-	DataHandlerDomain0 (class in bbc1.core.data_handler),
tribute), 36	30  Dh Adonton (class in bhol core data handlar) 22
compromised_asset_files (bbc1.core.message_key_types.KeyType	DbAdaptor (class in bbc1.core.data_handler), 32 deactivate() (bbc1.core.query_management.QueryEntry
attribute), 36	method), 39
compromised_transaction_data	del_entry() (bbc1.core.query_management.Ticker
(bbc1.core.message_key_types.KeyType	method), 39
attribute), 36	derive_shared_key() (in module
compromised_transactions	bbc1.core.message_key_types), 38
(bbc1.core.message_key_types.KeyType	deserialize() (bbc1.core.bbclib.BBcAsset method), 18
attribute), 36	deserialize() (bbc1.core.bbclib.BBcCrossRef method), 19
CONFIRM_KEY_EXCHANGE	deserialize() (bbc1.core.bbclib.BBcEvent method), 19
(bbc1.core.bbc_network.BBcNetwork at-	deserialize() (bbc1.core.bbclib.BBcPointer method), 20

deserialize() (bbc1.core.bbclib.BBcReference method),	36
20	ECDSA_SECP256k1 (bbc1.core.bbclib.KeyType at-
deserialize() (bbc1.core.bbclib.BBcRelation method), 20	tribute), 23
deserialize() (bbc1.core.bbclib.BBcSignature method), 21	exchange_key() (bbc1.core.bbc_app.BBcAppClient
deserialize() (bbc1.core.bbclib.BBcTransaction method),	method), 2
21	exec_sql() (bbc1.core.data_handler.DataHandler
deserialize() (bbc1.core.bbclib.BBcWitness method), 22	method), 28
deserialize_data() (in module	exec_sql() (bbc1.core.data_handler.DataHandlerDomain0
bbc1.core.message_key_types), 38	method), 30
destination_node_id (bbc1.core.message_key_types.KeyTy	
attribute), 36	method), 40
destination_user_id (bbc1.core.message_key_types.KeyTyp	**
attribute), 36	attribute), 36
destination_user_ids (bbc1.core.message_key_types.KeyTy	
attribute), 36	attribute), 36
digest() (bbc1.core.bbclib.BBcAsset method), 19	
digest() (bbc1.core.bbclib.BBcTransaction method), 22	F
direct_send_to_user() (in module	forwarding_list (bbc1.core.message_key_types.KeyType
bbc1.core.user_message_routing), 43	attribute), 36
direction (bbc1.core.message_key_types.KeyType	utariouto), 50
attribute), 36	G
dispatch() (bbc1.core.bbc_app.Callback method), 8	gather_signatures() (bbc1.core.bbc_app.BBcAppClient
DISTRIBUTE_CROSS_REF	method), 2
(bbc1.core.domain0_manager.Domain0Manager	
attribute), 33	generate() (bbc1.core.bbclib.KeyPair method), 23 generate_keypair() (bbc1.core.libbbcsig.pybbcsig.PyBBcSigSecp256k1
distribute_cross_ref_in_domain0()	method), 1
(bbc1.core.domain0_manager.Domain0Manager	get_asset_file() (bbc1.core.bbclib.BBcAsset method), 19
method), 34	get_bbc_config() (bbc1.core.bbc_app.BBcAppClient
Domain0Manager (class in	method), 3
bbc1.core.domain0_manager), 33	get_bigint() (in module bbc1.core.bbclib), 25
DOMAIN_ACCEPTANCE_RECOVER_INTERVAL	get_config() (bbc1.core.bbc_config.BBcConfig method),
(bbc1.core.domain0_manager.Domain0Manager	get_coning() (obet.core.obe_coning.bbeconing inteniou),
attribute), 33	get_destinations() (bbc1.core.bbclib.BBcReference
domain_close() (bbc1.core.bbc_app.BBcAppClient	method), 20
method), 2	get_domain_config() (bbc1.core.bbc_config.BBcConfig
domain_id (bbc1.core.message_key_types.KeyType at-	method), 12
tribute), 36	get_domain_keypair() (bbc1.core.bbc_network.BBcNetwork
DOMAIN_INFO_ADVERTISE_INTERVAL	method), 15
(bbc1.core.domain0_manager.Domain0Manager	get_domain_list() (bbc1.core.bbc_app.BBcAppClient
attribute), 33	method), 3
DOMAIN_INFO_LIFETIME	get_domain_neighborlist()
(bbc1.core.domain0_manager.Domain0Manager	(bbc1.core.bbc_app.BBcAppClient method), 3
attribute), 33	get_ECDH_parameters() (in module
domain_list (bbc1.core.message_key_types.KeyType at-	bbc1.core.message_key_types), 38
tribute), 36	get_entry() (bbc1.core.query_management.Ticker
DOMAIN_PING (bbc1.core.bbclib.MsgType attribute),	method), 39
23	<pre>get_forwarding_list() (bbc1.core.bbc_app.BBcAppClient</pre>
domain_ping (bbc1.core.message_key_types.KeyType at-	method), 3
tribute), 36	<pre>get_from_queue() (bbc1.core.bbc_app.Callback method),</pre>
domain_setup() (bbc1.core.bbc_app.BBcAppClient	8
method), 2	get_in_storage() (bbc1.core.data_handler.DataHandler
E	method), 28
	get_in_storage() (bbc1.core.data_handler.DataHandlerDomain0
ecdh (bbc1.core.message_key_types.KeyType attribute),	method), 30

get_json_config() (bb method), 12	oc1.core.bbc_config.BBcConfig	infra_msg_type (bbc1.core.message_key_types.KeyType attribute), 36
get_logger() (in module bl	bc1.core.logger), 35	InfraMessageCategory (class in
get_n_byte_int() (in modu		bbc1.core.message_key_types), 35
get_n_bytes() (in module 1		INITIAL_ACCEPT_LIMIT
get_new_id() (in module b		(bbc1.core.domain0_manager.Domain0Manager
_	c1.core.bbc_app.BBcAppClient	attribute), 33
method), 3	- <b>11</b>	insert_cross_ref() (bbc1.core.data_handler.DataHandler
	oc1.core.bbc_network.NodeInfo	method), 28
method), 17		insert_transaction() (bbc1.core.bbc_app.BBcAppClient
get_notification_list() (bbc	c1.core.bbc_app.BBcAppClient	method), 4
method), 3		<pre>insert_transaction() (bbc1.core.bbc_core.BBcCoreService</pre>
<pre>get_private_key_in_der()</pre>	(bbc1.core.bbclib.KeyPair	method), 13
method), 23		insert_transaction() (bbc1.core.data_handler.DataHandler
get_private_key_in_pem()	(bbc1.core.bbclib.KeyPair	method), 28
method), 23		insert_transaction() (bbc1.core.data_handler.DataHandlerDomain0
get_public_key_compress	ed()	method), 30
(bbc1.core.libbb	csig.pybbcsig.PyBBcSigSecp256	6kplv4_address (bbc1.core.message_key_types.KeyType at-
method), 1		tribute), 37
get_public_key_uncompre	essed()	ipv6_address (bbc1.core.message_key_types.KeyType at-
(bbc1.core.libbb	ocsig.pybbcsig.PyBBcSigSecp256	5k1 tribute), 37
method), 1		is_anycast (bbc1.core.message_key_types.KeyType at-
get_random_id() (in modu	ıle bbc1.core.bbclib), 26	tribute), 37
get_random_value() (in m	odule bbc1.core.bbclib), 26	is_less_than() (in module bbc1.core.bbc_network), 18
get_referred_transaction()		is_replication (bbc1.core.message_key_types.KeyType
(bbc1.core.bbcli	b.BBcReference method),	attribute), 37
20		ī
	oc1.core.bbclib.BBcTransaction	J
method), 22		JOIN_MULTICAST_RECEIVER
	_app.BBcAppClient method), 3	(bbc1.core.user_message_routing.UserMessageRouting
	_stats.BBcStats method), 18	attribute), 41
get_ticker() (in module	bbc1.core.query_management),	jsondump() (bbc1.core.bbclib.BBcTransaction method),
40		22
	c1.core.bbc_app.BBcAppClient	jsonload() (bbc1.core.bbclib.BBcTransaction method), 22
method), 4		
Н		K
		KEY_EXCHANGE_INVOKE_MAX_BACKOFF
HEADER_LEN (bbc1.com attribute), 38	re.message_key_types.Message	(bbc1.core.key_exchange_manager.KeyExchangeManager attribute), 34
	_key_types.KeyType attribute),	KEY_EXCHANGE_RETRY_INTERVAL
36	. 3=31 3 31	(bbc1.core.key_exchange_manager.KeyExchangeManager
	essage_key_types.KeyType at-	attribute), 34
tribute), 36		KEY_OBSOLETE_TIMER
		(bbc1.core.key_exchange_manager.KeyExchangeManager
		attribute), 34
include admin info() (bbc	c1.core.bbc_app.BBcAppClient	KEY_REFRESH_INTERVAL
method), 4	, in cross con _upp. DD or ipp on one	(bbc1.core.key_exchange_manager.KeyExchangeManager
include_admin_info_into_	message if needed()	attribute), 34
	network.BBcNetwork	KeyExchangeManager (class in
method), 16		bbc1.core.key_exchange_manager), 34
	c1.core.bbc_app.BBcAppClient	KeyPair (class in bbc1.core.bbclib), 23
method), 4		KeyType (class in bbc1.core.bbclib), 23
	re.message_key_types.KeyType	KeyType (class in bbc1.core.message_key_types), 36
attribute), 36		7 71 - (

L	node_id (bbc1.core.message_key_types.KeyType at-
LEAVE_MULTICAST_RECEIVER	tribute), 37
(bbc1.core.user_message_routing.UserMessageRattribute) 41	Routing info (bbc1.core.message_key_types.KeyType attribute), 37
ledger_subsys_manip (bbc1.core.message_key_types.KeyTattribute), 37	NODEINFO_LIFETIME
ledger_subsys_register (bbc1.core.message_key_types.Key_attribute) 37	tribute), 17
ledger_subsys_verify (bbc1.core.message_key_types.KeyT attribute), 37	ypedekey_signature (bbc1.core.message_key_types.KeyType attribute), 37
M	nonce (bbc1.core.message_key_types.KeyType attribute), 37
make_binary() (in module	notification_list (bbc1.core.message_key_types.KeyType
bbc1.core.message_key_types), 38	attribute), 37
make_dictionary_from_TLV_format() (in module bbc1.core.message_key_types), 38	NOTIFY_CROSS_REF (bbc1.core.bbclib.MsgType attribute), 23
make_message() (in module	NOTIFY_CROSS_REF_REGISTERED
bbc1.core.message_key_types), 38	(bbc1.core.domain0_manager.Domain0Manager
make_neighbor_list() (bbc1.core.topology_manager.Topolomethod), 41	NOTH I_DOMAIN_KLI_CIDATE
$make\_TLV\_formatted\_message() \hspace{1cm} (in \hspace{1cm} module$	(bbc1.core.bbclib.MsgType attribute), 23
bbc1.core.message_key_types), 38	notify_domain_key_update()  (bbal agra bba ann BBa Ann Client method) 4
make_transaction() (in module bbc1.core.bbclib), 26	(bbc1.core.bbc_app.BBcAppClient method), 4 NOTIFY_INSERTED (bbc1.core.bbclib.MsgType
make_transaction_for_base_asset() (in module	attribute), 23
bbc1.core.bbclib), 26	NOTIFY_INSERTED (bbc1.core.data_handler.DataHandler
manipulate_ledger_subsystem() (bbc1.core.bbc_app.BBcAppClient method), 4	attribute), 27
MAX_CROSS_REF_STOCK	NOTIFY_LEAVE (bbc1.core.bbc_network.BBcNetwork
(bbc1.core.user_message_routing.UserMessageR	Routing attribute), 15
attribute), 41	NOTIFY_NEIGHBOR_LIST
merkle_tree (bbc1.core.message_key_types.KeyType at-	$(bbc1.core.topology\_manager.TopologyManagerBase\\$
tribute), 37	attribute), 41
MESSAGE (bbc1.core.bbclib.MsgType attribute), 23	notify_neighbor_update()
message (bbc1.core.message_key_types.KeyType at-	(bbc1.core.topology_manager.TopologyManagerBase
tribute), 37	method), 41 NUM_OF_COPIES (bbc1.core.domain0_manager.Domain0Manager
Message (class in bbc1.core.message_key_types), 38	attribute), 33
message_seq (bbc1.core.message_key_types.KeyType attribute), 37	O
mk_keyobj_from_private_key()	
(bbc1.core.bbclib.KeyPair method), 23	on_multinodes (bbc1.core.message_key_types.KeyType attribute), 37
mk_keyobj_from_private_key_der() (bbc1.core.bbclib.KeyPair method), 23	open_db() (bbc1.core.data_handler.DbAdaptor method),
mk_keyobj_from_private_key_pem()	32
(bbc1.core.bbclib.KeyPair method), 23	open_db() (bbc1.core.data_handler.MysqlAdaptor
MsgType (class in bbc1.core.bbclib), 23	method), 32
MysqlAdaptor (class in bbc1.core.data_handler), 32	open_db() (bbc1.core.data_handler.SqliteAdaptor method), 33
N	outer_domain_id (bbc1.core.message_key_types.KeyType
neighbor_list (bbc1.core.message_key_types.KeyType at-	attribute), 37
tribute), 37	output_der() (bbc1.core.libbbcsig.pybbcsig.PyBBcSigSecp256k1
NEIGHBOR LIST REFRESH INTERVAL	method), 1
(bbc1.core.topology_manager.TopologyManager attribute), 41	Butput_pem() (bbc1.core.libbbcsig.pybbcsig.PyBBcSigSecp256k1 method), 1
NeighborInfo (class in bbc1.core.bbc_network), 17	

P	method), 11
<pre>parse() (bbc1.core.message_key_types.Message method),</pre>	proc_resp_traverse_transactions()
38	(bbc1.core.bbc_app.Callback method), 11
parser() (in module bbc1.core.command), 27	proc_resp_verify_cross_ref()
PayloadType (class in bbc1.core.message_key_types), 38	(bbc1.core.bbc_app.Callback method), 11
port_number (bbc1.core.message_key_types.KeyType at-	proc_resp_verify_hash() (bbc1.core.bbc_app.Callback method), 11
tribute), 37	proc_user_message() (bbc1.core.bbc_app.Callback
prepare_reference() (bbc1.core.bbclib.BBcReference method), 20	method), 11
$\begin{array}{ccc} proc\_cmd\_sign\_request() & (bbc1.core.bbc\_app.Callback\\ & method),  8 \end{array}$	process_message() (bbc1.core.data_handler.DataHandler method), 29
proc_notify_cross_ref() (bbc1.core.bbc_app.Callback method), 8	process_message() (bbc1.core.data_handler.DataHandlerDomain0 method), 31
proc_notify_inserted() (bbc1.core.bbc_app.Callback method), 9	process_message() (bbc1.core.domain0_manager.Domain0Manager method), 34
proc_resp_cross_ref_list() (bbc1.core.bbc_app.Callback method), 9	process_message() (bbc1.core.topology_manager.TopologyManagerBase method), 41
proc_resp_domain_close() (bbc1.core.bbc_app.Callback method), 9	process_message() (bbc1.core.user_message_routing.UserMessageRouting method), 42
proc_resp_domain_setup() (bbc1.core.bbc_app.Callback method), 9	process_message() (bbc1.core.user_message_routing.UserMessageRouting method), 42
proc_resp_ecdh_key_exchange() (bbc1.core.bbc_app.Callback method), 9	purge() (bbc1.core.bbc_network.NeighborInfo method), 17
proc_resp_gather_signature()	PURGE_INTERVAL_SEC
(bbc1.core.bbc_app.Callback method), 9	(bbc1.core.bbc_network.NeighborInfo at-
proc_resp_get_config() (bbc1.core.bbc_app.Callback	tribute), 17
method), 9	put_message() (bbc1.core.repair_manager.RepairManager
proc_resp_get_domainlist() (bbc1.core.bbc_app.Callback method), 9	method), 40 PyBBcSigSecp256k1 (class in
proc_resp_get_forwardinglist()	bbc1.core.libbbcsig.pybbcsig), 1
(bbc1.core.bbc_app.Callback method), 9	Q
proc_resp_get_neighborlist()	
(bbc1.core.bbc_app.Callback method), 9 proc_resp_get_node_id() (bbc1.core.bbc_app.Callback	query_id (bbc1.core.message_key_types.KeyType attribute), 37
method), 10	QueryEntry (class in bbc1.core.query_management), 39
proc_resp_get_notificationlist()	quit_program() (bbc1.core.bbc_core.BBcCoreService
(bbc1.core.bbc_app.Callback method), 10	method), 13
proc_resp_get_stats() (bbc1.core.bbc_app.Callback method), 10	R
$proc\_resp\_get\_userlist() \qquad (bbc1.core.bbc\_app.Callback$	random (bbc1.core.message_key_types.KeyType at-
method), 10 proc_resp_insert() (bbc1.core.bbc_app.Callback method),	tribute), 37 read_config() (bbc1.core.bbc_config.BBcConfig
10	method), 12
proc_resp_ledger_subsystem() (bbc1.core.bbc_app.Callback method), 10	reason (bbc1.core.message_key_types.KeyType attribute), 37
proc_resp_register_hash() (bbc1.core.bbc_app.Callback method), 10	receive_confirmation() (bbc1.core.key_exchange_manager.KeyExchangeM method), 34
proc_resp_search_transaction()	receive_exchange_request()
(bbc1.core.bbc_app.Callback method), 10	(bbc1.core.key_exchange_manager.KeyExchangeManager
proc_resp_search_with_condition()	method), 34
(bbc1.core.bbc_app.Callback method), 10	receive_exchange_response()
proc_resp_set_neighbor() (bbc1.core.bbc_app.Callback method), 11	(bbc1.core.key_exchange_manager.KeyExchangeManager
proc_resp_sign_request() (bbc1.core.bbc_app.Callback	method), 35

receiver_loop()	(bbc1.core.bbc_app.BBcAppClient	REQUEST_CLOSE_DOMAIN	
method), 4		(bbc1.core.bbclib.MsgType attribute), 24	
recover_asset_file()	(bbc1.core.bbclib.BBcAsset	request_cross_ref_holders_list()	
method), 19	9	(bbc1.core.bbc_app.BBcAppClient method), 5	5
	<pre>pject() (in module bbc1.core.bbclib),</pre>	REQUEST_CROSS_REF_LIST	
- & - 26	,,	(bbc1.core.bbclib.MsgType attribute), 24	
	essage_key_types.Message method),	REQUEST_CROSS_REF_VERIFY	
38	ssage_nej_ejpesimessage meanes),	(bbc1.core.bbclib.MsgType attribute), 24	
	1.core.message_key_types.KeyType	REQUEST_ECDH_KEY_EXCHANGE	
attribute), 3		(bbc1.core.bbclib.MsgType attribute), 24	
, ,	RDING_LIST_INTERVAL	REQUEST_GATHER_SIGNATURE	
	user_message_routing.UserMessageR	• = =	
attribute), 4		REQUEST_GET_CONFIG (bbc1.core.bbclib.MsgTyp	
	re.bbclib.MsgType attribute), 23		,
		attribute), 24	
register_in_ledger_su	•	REQUEST_GET_DOMAINLIST	
	bbc_app.BBcAppClient method), 4	(bbc1.core.bbclib.MsgType attribute), 24	
register_to_core()	(bbc1.core.bbc_app.BBcAppClient	REQUEST_GET_FORWARDING_LIST	
method), 5		(bbc1.core.bbclib.MsgType attribute), 24	
		SAR ERMIEST_GET_NEIGHBORLIST	
method), 42		(bbc1.core.bbclib.MsgType attribute), 24	
		salgeRuiEngDGinTinyNODEID (bbc1.core.bbclib.MsgTyp	e
method), 43		attribute), 24	
	bc1.core.bbc_network.NeighborInfo	REQUEST_GET_NOTIFICATION_LIST	
method), 1'	7	(bbc1.core.bbclib.MsgType attribute), 24	
remove() (bbc1.core.	data_handler.DataHandler method),	REQUEST_GET_STATS (bbc1.core.bbclib.MsgType at	t-
29		tribute), 24	
remove() (bbc1.core	.data_handler.DataHandlerDomain0	REQUEST_GET_USERS (bbc1.core.bbclib.MsgType at	t-
method), 3	1	tribute), 24	
remove_domain() (b	bbc1.core.bbc_network.BBcNetwork	REQUEST_INSERT (bbc1.core.bbclib.MsgType at	t-
method), 10		tribute), 24	
remove_domain_con		request_insert_completion_notification()	
	bbc_config.BBcConfig method), 12	(bbc1.core.bbc_app.BBcAppClient method), 5	5
remove_from_notific		REQUEST_INSERT_NOTIFICATION	
	bbc_core.BBcCoreService method),	(bbc1.core.bbclib.MsgType attribute), 24	
13	boe_eore.bbecoreservice memoay,	REQUEST_KEY_EXCHANGE	
remove_old_key()	(in module	(bbc1.core.bbc_network.BBcNetwork at	t_
	key_exchange_manager), 35	tribute), 15	L
		REQUEST_MANIP_LEDGER_SUBSYS	
method), 18		(bbc1.core.bbclib.MsgType attribute), 24	
remove stat item()	(bbc1.core.bbc stats.BBcStats	REQUEST REGISTER HASH IN SUBSYS	
'	` =	`	
method), 18		(bbc1.core.bbclib.MsgType attribute), 24	
REPAIR_TRANSAC		REQUEST_REPAIR (bbc1.core.bbclib.MsgType at	t-
,	data_handler.DataHandler at-	tribute), 24	
tribute), 27		REQUEST_REPAIR_ASSET_FILE	
-	s in bbc1.core.repair_manager), 40	(bbc1.core.repair_manager.RepairManager	
	L (bbc1.core.data_handler.DataHandle		
attribute), 2		REQUEST_REPAIR_TRANSACTION	
REPLICATION_CRO		(bbc1.core.repair_manager.RepairManager	
	data_handler.DataHandler at-	attribute), 40	
tribute), 27		REQUEST_REPLICATION_INSERT	
REPLICATION_EXT	$\Gamma$ (bbc1.core.data_handler.DataHandle	er (bbc1.core.data_handler.DataHandler at	t-
attribute), 2	27	tribute), 28	
REPLICATION_P2P	(bbc1.core.data_handler.DataHandler	r REQUEST_SEARCH (bbc1.core.data_handler.DataHan	ıdler
attribute), 2		attribute), 28	

RESPONSE_GET_FORWARDING_LIST
(bbc1.core.bbclib.MsgType attribute), 24
RESPONSE_GET_NEIGHBORLIST
(bbc1.core.bbclib.MsgType attribute), 24
RESPONSE_GET_NODEID (bbc1.core.bbclib.MsgType
attribute), 24
RESPONSE_GET_NOTIFICATION_LIST
(bbc1.core.bbclib.MsgType attribute), 24
RESPONSE_GET_STATS (bbc1.core.bbclib.MsgType
attribute), 25
RESPONSE_GET_USERS (bbc1.core.bbclib.MsgType
attribute), 25
RESPONSE_INSERT (bbc1.core.bbclib.MsgType
attribute), 25
RESPONSE_KEY_EXCHANGE
(bbc1.core.bbc_network.BBcNetwork at-
tribute), 15
RESPONSE_MANIP_LEDGER_SUBSYS
(bbc1.core.bbclib.MsgType attribute), 25
RESPONSE_NO_SUCH_USER
(bbc1.core.user_message_routing.UserMessageRouting
attribute), 41
nRRSnPOSSE_REGISTER_HASH_IN_SUBSYS
(bbc1.core.bbclib.MsgType attribute), 25
RESPONSE_REPLICATION_INSERT
(bbc1.core.data_handler.DataHandler at-
tribute), 28
RESPONSE_SEARCH (bbc1.core.data_handler.DataHandler
attribute), 28
RESPONSE_SEARCH_TRANSACTION
(bbc1.core.bbclib.MsgType attribute), 25
RESPONSE_SEARCH_WITH_CONDITIONS
JserMessag <b>&amp;Routing</b> e.bbclib.MsgType attribute), 25
RESPONSE_SET_STATIC_NODE
(bbc1.core.bbclib.MsgType attribute), 25
RESPONSE_SETUP_DOMAIN
(bbc1.core.bbclib.MsgType attribute), 25
RESPONSE_SIGNATURE (bbc1.core.bbclib.MsgType
attribute), 25
RESPONSE_TRANSACTION_DATA
(bbc1.core.repair_manager.RepairManager
attribute), 40
RESPONSE_TRAVERSE_TRANSACTIONS
(bbc1.core.bbclib.MsgType attribute), 25
RESPONSE_USER_LOCATION
(bbc1.core.user_message_routing.UserMessageRouting
attribute), 42
RESPONSE_VERIFY_FROM_OUTER_DOMAIN
(bbc1.core.domain0_manager.Domain0Manager
attribute), 33
RESPONSE_VERIFY_HASH_IN_SUBSYS
(bbc1.core.bbclib.MsgType attribute), 25
restore_transaction_data()
(bbc1.core.data_handler.DataHandler method),

29	send_message_in_network()
result (bbc1.core.message_key_types.KeyType attribute),	(bbc1.core.bbc_network.BBcNetwork
37	method), 16
retry_timer (bbc1.core.message_key_types.KeyType attribute), 37	send_message_to_a_domain0_manager() (bbc1.core.bbc_network.BBcNetwork
tribute), 37	method), 16
S	send_message_to_user() (bbc1.core.user_message_routing.UserMessageRo
	method), 42
save_all_static_node_list() (bbc1.core.bbc_network.BBcNetwork	send_message_to_user() (bbc1.core.user_message_routing.UserMessageRo
method), 16	method), 43
search_domain_having_cross_ref()	send_multicast_join() (bbc1.core.user_message_routing.UserMessageRouti
(bbc1.core.data_handler.DataHandler method),	method), 42
29	send_multicast_join() (bbc1.core.user_message_routing.UserMessageRouti
search_transaction() (bbc1.core.bbc_app.BBcAppClient	method), 43
method), 5	send_multicast_leave() (bbc1.core.user_message_routing.UserMessageRou
search_transaction() (bbc1.core.data_handler.DataHandler	method), 42
method), 29	sendback_denial_of_sign()
search_transaction() (bbc1.core.data_handler.DataHandlerI	Domain() (bbc1.core.bbc_app.BBcAppClient method), 6
method), 31	sendback_signature() (bbc1.core.bbc_app.BBcAppClient
search_transaction_topology()	method), 7
(bbc1.core.data_handler.DataHandler method),	serialize() (bbc1.core.bbclib.BBcAsset method), 19
29	serialize() (bbc1.core.bbclib.BBcCrossRef method), 19
search_transaction_topology()	serialize() (bbc1.core.bbclib.BBcEvent method), 19
(bbc1.core.data_handler.DataHandlerDomain0	serialize() (bbc1.core.bbclib.BBcPointer method), 20
method), 31	serialize() (bbc1.core.bbclib.BBcReference method), 20
search_transaction_with_condition()	serialize() (bbc1.core.bbclib.BBcRelation method), 21
(bbc1.core.bbc_app.BBcAppClient method), 5	serialize() (bbc1.core.bbclib.BBcSignature method), 21
search_transaction_with_condition()	serialize() (bbc1.core.bbclib.BBcTransaction method), 22
(bbc1.core.bbc_core.BBcCoreService method),	serialize() (bbc1.core.bbclib.BBcWitness method), 23
13	set_aes_name() (bbc1.core.user_message_routing.UserMessageRouting
SECURITY_STATE_CONFIRMING	method), 42 set_callback() (bbc1.core.bbc_app.BBcAppClient
(bbc1.core.bbc_network.NodeInfo attribute),	method), 7
17 SECURITY_STATE_ESTABLISHED	set_cipher() (bbc1.core.key_exchange_manager.KeyExchangeManager
(bbc1.core.bbc_network.NodeInfo attribute),	method), 35
17	set_cipher() (in module bbc1.core.message_key_types),
SECURITY_STATE_NONE	38
(bbc1.core.bbc_network.NodeInfo attribute),	set_client() (bbc1.core.bbc_app.Callback method), 11
17	set_domain_id() (bbc1.core.bbc_app.BBcAppClient
SECURITY_STATE_REQUESTING	method), 7
(bbc1.core.bbc_network.NodeInfo attribute),	set_domain_static_node()
17	(bbc1.core.bbc_app.BBcAppClient method), 7
send_domain_ping() (bbc1.core.bbc_app.BBcAppClient	set_error() (in module bbc1.core.bbclib), 26
method), 6	$set\_invoke\_timer() (bbc1.core.key\_exchange\_manager.KeyExchangeManager.KeyExchange.KeyExc$
send_domain_ping() (bbc1.core.bbc_network.BBcNetwork	
method), 16	set_keypair() (bbc1.core.bbc_app.BBcAppClient
send_inserted_notification()	method), 7
(bbc1.core.bbc_core.BBcCoreService method),	set_logger() (bbc1.core.bbc_app.Callback method), 11
13	set_node_key() (bbc1.core.bbc_app.BBcAppClient
send_key_exchange_message()	method), 7
(bbc1.core.bbc_network.BBcNetwork	set_user_id() (bbc1.core.bbc_app.BBcAppClient
method), 16	method), 7
send_message() (bbc1.core.bbc_app.BBcAppClient	setup_tcp_server() (bbc1.core.bbc_network.BBcNetwork
method), 6	method), 17

	<del>-</del>
T =	Т
method), 17	tcpserver_loop() (bbc1.core.bbc_network.BBcNetwork
show_list() (bbc1.core.bbc_network.NeighborInfo	method), 17
method), 17	Ticker (class in bbc1.core.query_management), 39
sign() (bbc1.core.bbclib.BBcTransaction method), 22	to_1byte() (in module bbc1.core.bbclib), 26
sign() (bbc1.core.bbclib.KeyPair method), 23	to_2byte() (in module bbc1.core.bbclib), 26
sign() (bbc1.core.libbbcsig.pybbcsig.PyBBcSigSecp256k1	to_2byte() (in module bbc1.core.message_key_types), 38
method), 1	to_4byte() (in module bbc1.core.bbclib), 26
signature (bbc1.core.message_key_types.KeyType	to_4byte() (in module bbc1.core.message_key_types), 39
attribute), 37	to 8byte() (in module bbc1 core bbclib) 26
source_domain_id (bbc1.core.message_key_types.KeyType	to bigint() (bbc1.core.bbclib.KeyPair method), 23
attribute), 37	to_bigint() (in module bbc1.core.bbclib), 26
source_node_id (bbc1.core.message_key_types.KeyType	to_binary() (bbc1.core.bbclib.KeyPair method), 23
attribute), 37	TopologyManagerBase (class in
source_user_id (bbc1.core.message_key_types.KeyType	bbc1.core.topology_manager), 40
attribute), 37	touch() (bbc1.core.bbc_network.NodeInfo method), 17
SqliteAdaptor (class in bbc1.core.data_handler), 32	transaction_data (bbc1.core.message_key_types.KeyType
start_receiver_loop() (bbc1.core.bbc_app.BBcAppClient	attribute), 37
method), 8	transaction_id (bbc1.core.message_key_types.KeyType
STATE_CONFIRMING (bbc1.core.key_exchange_manage	r.KeyExchangeManager
attribute), 34	transaction_id_list (bbc1.core.message_key_types.KeyType
STATE_ESTABLISHED	attribute), 37
(bbc1.core.key exchange manager.KeyExchange	Managefon_tree (bbc1.core.message_key_types.KeyType
attribute), 34	attribute), 37
STATE_NONE (bbc1.core.key_exchange_manager.KeyExc	Change Manager had some massage leavy turnes VayTurne at
attribute), 34	tribute), 38
STATE REOUESTING (bbc1.core.kev exchange manage	rtKevEsc_hrassactabaser(bbc1.core.bbc_app.BBcAppClient
attribute), 34	
static_entry (bbc1.core.message_key_types.KeyType at-	method), 8
tribute), 37	txid_having_cross_ref (bbc1.core.message_key_types.KeyType
stats (bbc1.core.message_key_types.KeyType attribute),	attribute), 38
37	Type_any (bbc1.core.message_key_types.PayloadType
status (bbc1.core.message_key_types.KeyType attribute),	attribute), 38
37	Type_binary (bbc1.core.message_key_types.PayloadType
stop_all_timers() (bbc1.core.domain0_manager.Domain0M	attribute), 38
method), 34	
stop_all_timers() (bbc1.core.key_exchange_manager.KeyE	(bbc1.core.message_key_types.PayloadType
method), 35	
stop_all_timers() (bbc1.core.topology_manager.TopologyM	Type_msgpack (bbc1.core.message_key_types.PayloadType
method), 41	allager Basattribute), 38
stop_all_timers() (bbc1.core.user_message_routing.UserMe	adshaa Pauting
method), 42	
step all times () (bbal some year massage routing UserMa	udp_message_loop() (bbc1.core.bbc_network.BBcNetwork
stop_all_timers() (bbc1.core.user_message_routing.UserMe	essage Routing Dulanty
method), 43	UNREGISTER (bbc1.core.bbclib.MsgType attribute), 25
store_in_storage() (bbc1.core.data_handler.DataHandler	unregister_from_core() (bbc1.core.bbc_app.BBcAppClient
method), 30	method), 8
store_in_storage() (bbc1.core.data_nandier.DataHandierDo	method), 8 main() unregister_user() (bbc1.core.user_message_routing.UserMessageRouting
1110 (1110 (1)), 2 1	method), 42
str_binary() (in module bbc1.core.bbclib), 26	unregister_user() (bbc1.core.user_message_routing.UserMessageRoutingDescription
sync_by_queryid() (bbc1.core.bbc_app.Callback	method), 43
method), 11	unset_cipher() (bbc1.core.key_exchange_manager.KeyExchangeManager
synchronize() (bbc1.core.bbc_app.Callback method), 12	method), 35
	unset_cipher() (in module
	bbc1.core.message key types), 39

```
update() (bbc1.core.bbc network.NodeInfo method), 17
             (bbc1.core.query_management.QueryEntry
update()
         method), 39
update_config()
                     (bbc1.core.bbc_config.BBcConfig
         method), 12
update deep() (in module bbc1.core.bbc config), 12
update domain belong to()
         (bbc1.core.domain0 manager.Domain0Manager
         method), 34
update_expiration_time()
         (bbc1.core.query_management.QueryEntry
         method), 39
update_refresh_timer_entry()
         (bbc1.core.topology_manager.TopologyManagerBase
         method), 41
update_stats() (bbc1.core.bbc_stats.BBcStats method), 18
update_stats_decrement() (bbc1.core.bbc_stats.BBcStats
         method), 18
update_stats_increment() (bbc1.core.bbc_stats.BBcStats
         method), 18
user_id
         (bbc1.core.message_key_types.KeyType
                                                   at-
         tribute), 38
user_list (bbc1.core.message_key_types.KeyType
                                                   at-
         tribute), 38
UserMessageRouting
                                 (class
                                                   in
         bbc1.core.user_message_routing), 41
UserMessageRoutingDummy
                                     (class
                                                   in
         bbc1.core.user_message_routing), 42
V
validate_transaction() (bbc1.core.bbc_core.BBcCoreService
         method), 14
validate_transaction_object()
                                   (in
                                              module
         bbc1.core.bbclib), 26
verify() (bbc1.core.bbclib.BBcSignature method), 21
verify() (bbc1.core.bbclib.KeyPair method), 23
verify() (bbc1.core.libbbcsig.pybbcsig.PyBBcSigSecp256k1
         method), 1
verify in ledger subsystem()
         (bbc1.core.bbc_app.BBcAppClient method), 8
verify using cross ref() (in module bbc1.core.bbclib),
         26
```