

---

# **BBc1 Documentation**

*Release 1.4*

**beyond-blockchain.org**

**May 10, 2020**



---

## Contents:

---

<b>1</b>	<b>bbc1 package</b>	<b>1</b>
1.1	Subpackages . . . . .	1
1.1.1	bbc1.core package . . . . .	1
1.1.1.1	Submodules . . . . .	1
1.1.1.2	Module contents . . . . .	37
1.2	Module contents . . . . .	37
<b>2</b>	<b>Indices and tables</b>	<b>39</b>
	<b>Python Module Index</b>	<b>41</b>
	<b>Index</b>	<b>43</b>



## 1.1 Subpackages

### 1.1.1 `bbc1.core` package

#### 1.1.1.1 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.

```
class bbc1.core.bbc_app.BBcAppClient (host='127.0.0.1', port=9000, multiq=True,  
logname='-', loglevel='none')
```

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`

**count\_transactions** (*asset\_group\_id=None, asset\_id=None, user\_id=None, start\_from=None, until=None*)

Count transactions that matches the given conditions

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
- **start\_from** (*int*) – the starting timestamp to search
- **until** (*int*) – the end timestamp to search

**Returns** the number of transactions

**Return type** int

**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, anycast=False*)

Request to gather signatures from the specified user\_ids

**Parameters**

- **txobj** (*BBcTransaction*) –
- **reference\_obj** (*BBcReference*) – BBcReference object that includes the information 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** (*txobj*)

Request to insert a legitimate transaction

**Parameters** *txobj* (*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

**search\_transaction\_with\_condition** (*asset\_group\_id=None, asset\_id=None, user\_id=None, start\_from=None, until=None, direction=0, count=0*)

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

- **start\_from** (*int*) – the starting timestamp to search
- **until** (*int*) – the end timestamp to search
- **direction** (*int*) – 0: descend, 1: ascend
- **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

**sendback\_signature** (*dest\_user\_id=None, transaction\_id=None, ref\_index=-1, signature=None, query\_id=None*)

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*, *asset\_group\_id=None*, *user\_id=None*,  
*start\_from=None, until=None, 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
- **asset\_group\_id** (*bytes*) – `asset_group_id` that target transactions should have
- **user\_id** (*bytes*) – `user_id` that target transactions should have
- **start\_from** (*int*) – the starting timestamp to search
- **until** (*int*) – the end timestamp to search
- **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\_count\_transactions** (*dat*)

Callback for message RESPONSE\_COUNT\_TRANSACTIONS

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, default\_confpath=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.load_config` (*filepath*)

`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,                core_port=None,
                                         use_domain0=False,         ip4addr=None,
                                         ip6addr=None,              workingdir='.bbc1',
                                         configfile=None,           use_nodekey=None,
                                         use_ledger_subsystem=False, default_confdir=None,
                                         default_confdir=None, loglevel='all', logname='-',
                                         server_start=True)
```

Bases: object

Base service object of BBc-1

```
count_transactions (domain_id,  asset_group_id=None,  asset_id=None,  user_id=None,
                    start_from=None, until=None)
```

Count 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
- **start\_from** (*int*) – the starting timestamp to search
- **until** (*int*) – the end timestamp to search

**Returns** the number of transactions**Return type** int

```
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 quitting 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\_transaction\_with\_condition** (*domain\_id, asset\_group\_id=None, asset\_id=None, user\_id=None, start\_from=None, until=None, direction=0, count=0*)

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
- **start\_from** (*int*) – the starting timestamp to search
- **until** (*int*) – the end timestamp to search
- **direction** (*int*) – 0: descend, 1: ascend
- **count** (*int*) – The maximum number of transactions to retrieve (if `count <= 0`, then `self.search_max_count` is applied)

**Returns** dictionary having `transaction_id`, serialized transaction data, asset files

**Return type** dict

**send\_inserted\_notification** (*domain\_id, asset\_group\_ids, transaction\_id, only\_registered\_user=False*)

Broadcast NOTIFY\_INSERTED

**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. int (short): 2-byte value of BBcFormat type or None

**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_ip6addr=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



- **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\_key\_exchange\_message** (*domain\_id, node\_id, command, pubkey, nonce, random\_val, key\_name*)

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
- **domain0** (*bool or None*) – If True, the node is domain0 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.

### 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.

```
class bbc1.core.data_handler.DataHandler (networking=None, config=None, workingdir=None, domain_id=None, loglevel='all', logname=None)
```

Bases: object

DB and storage handler

```
NOTIFY_INSERTED = b'\x00\x04'
```

```
REPAIR_TRANSACTION_DATA = b'\x00\x05'
```

```
REPLICATION_ALL = 0
```

```
REPLICATION_CROSS_REF = b'\x00\x06'
```

```
REPLICATION_EXT = 2
```

```
REPLICATION_P2P = 1
```

```
REQUEST_REPLICATION_INSERT = b'\x00\x00'
```

```
REQUEST_SEARCH = b'\x00\x02'
```

```
RESPONSE_REPLICATION_INSERT = b'\x00\x01'
```

```
RESPONSE_SEARCH = b'\x00\x03'
```

```
count_domain_in_cross_ref (outer_domain_id)
```

Count the number of domains in the cross\_ref table

```
count_transactions (asset_group_id=None, asset_id=None, user_id=None, start_from=None, until=None, db_num=0)
```

Count transactions that matches the given conditions

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

#### Parameters

- **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
- **start\_from** (*int*) – the starting timestamp to search
- **until** (*int*) – the end timestamp to search
- **db\_num** (*int*) – index of DB if multiple DBs are used

**Returns** the number of transactions

**Return type** int

```
exec_sql (db_num=0, sql=None, args=(), commit=False, fetch_one=False, return_cursor=False)
```

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
- **return\_cursor** (*bool*) – If True (and fetch\_one is False), return db\_cur (iterator)



**Returns** list of records

**Return type** list

**get\_asset\_info** (*txobj*)

Retrieve asset information from transaction object

**Parameters** **txobj** (*BBcTransaction*) – transaction object to analyze

**Returns** list of list [asset\_group\_id, asset\_id, user\_id, file\_size, file\_digest]

**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\_cross\_ref** (*transaction\_id, outer\_domain\_id, txid\_having\_cross\_ref, no\_replication=False*)

Insert cross\_ref information into cross\_ref\_table

**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, fmt\_type=0, 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
- **fmt\_type** (*int*) – 2-byte value of BBcFormat type
- **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** (*transaction\_id=None, asset\_group\_id=None, asset\_id=None, user\_id=None, start\_from=None, until=None, direction=0, count=1, db\_num=0*)

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
- **start\_from** (*int*) – the starting timestamp to search
- **until** (*int*) – the end timestamp to search
- **direction** (*int*) – 0: descend, 1: ascend
- **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

**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

```
class bbc1.core.data_handler.DataHandlerDomain0 (networking=None, config=None,
workingdir=None, domain_id=None,
loglevel='all', logname=None)
```

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
- **return\_cursor** (*bool*) – If True (and fetch\_one is False), return db\_cur (iterator)

**Returns** list of records

**Return type** list

```
get_asset_info (txobj)
Retrieve asset information from transaction object
```

**Parameters** *txobj* (*BBcTransaction*) – transaction object to analyze

**Returns** list of list [asset\_group\_id, asset\_id, user\_id, file\_size, file\_digest]

**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
- **fmt\_type** (*int*) – 2-byte value of BBcFormat type
- **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** (*transaction\_id=None, asset\_group\_id=None, asset\_id=None, user\_id=None, count=1*)

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
- **start\_from** (*int*) – the starting timestamp to search
- **until** (*int*) – the end timestamp to search
- **direction** (*int*) – 0: descend, 1: ascend
- **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

**create\_ver2\_column** ()

Create column for version 2 meta table (add timestamp in asset\_info\_table)

**get\_version** ()

get\_version of the DB

**Returns** version string

**Return type** str

**open\_db** ()

Open the DB

**update\_table\_def** (*from\_ver*)

Update table definition

**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"}, [{"column\_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"}, [{"column\_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'\x00\x00'`

**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'\x00\x04'
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_DOMAIN0 = b'\x00\x04'`

`CATEGORY_NETWORK = b'\x00\x00'`

`CATEGORY_TOPOLOGY = b'\x00\x01'`

`CATEGORY_USER = b'\x00\x02'`

**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'\x00\x00\x00h'`

`anycast_ttl = b'\x00\x00\x00\x1a'`

`asset_file = b'\x00\x00\x00t'`

`asset_group_id = b'\x00\x00\x00c'`

`asset_group_ids = b'\x00\x00\x00d'`

`asset_id = b'\x00\x00\x00e'`

`bbc_configuration = b'\x00\x00\x00<'`

`command = b'\x00\x00\x00t'`

`compromised_asset_files = b'\x00\x00\x00\x92'`

`compromised_transaction_data = b'\x00\x00\x00\x90'`

`compromised_transaction_ids = b'\x00\x00\x00\x93'`

`compromised_transactions = b'\x00\x00\x00\x91'`

`count = b'\x00\x00\x00\x0e'`

`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'\x00\x00\x00R'  
destination_user_ids = b'\x00\x00\x00S'  
direction = b'\x00\x00\x00f'  
domain_id = b'\x00\x00\x00P'  
domain_list = b'\x00\x00\x007'  
domain_ping = b'\x00\x00\x00\x15'  
ecdh = b'\x00\x00\x00\x11'  
external_ip4addr = b'\x00\x00\x004'  
external_ip6addr = b'\x00\x00\x005'  
forwarding_list = b'\x00\x00\x008'  
hint = b'\x00\x00\x00\x10'  
hop_count = b'\x00\x00\x00g'  
infra_command = b'\x00\x00\x00\n'  
infra_msg_type = b'\x00\x00\x00\x08'  
ipv4_address = b'\x00\x00\x001'  
ipv6_address = b'\x00\x00\x002'  
is_anycast = b'\x00\x00\x00\x19'  
is_replication = b'\x00\x00\x00\x1b'  
ledger_subsys_manip = b'\x00\x00\x00\xa0'  
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\x0c'  
message_seq = b'\x00\x00\x00\x14'  
neighbor_list = b'\x00\x00\x00:'  
node_id = b'\x00\x00\x00T'  
node_info = b'\x00\x00\x006'  
nodekey_signature = b'\x00\x00\x00\x16'  
nonce = b'\x00\x00\x00r'  
notification_list = b'\x00\x00\x00;'  
on_multinodes = b'\x00\x00\x00\x18'  
outer_domain_id = b'\x00\x00\x00x'  
port_number = b'\x00\x00\x003'  
query_id = b'\x00\x00\x00\x0b'  
random = b'\x00\x00\x00\x12'  
reason = b'\x00\x00\x00\x01'
```

```

ref_index = b'\x00\x00\x00s'
result = b'\x00\x00\x00\x02'
retry_timer = b'\x00\x00\x00\x13'
signature = b'\x00\x00\x00v'
source_domain_id = b'\x00\x00\x00y'
source_node_id = b'\x00\x00\x00U'
source_user_id = b'\x00\x00\x00Q'
start_from = b'\x00\x00\x00i'
static_entry = b'\x00\x00\x000'
stats = b'\x00\x00\x00\x0f'
status = b'\x00\x00\x00\x00'
transaction_data = b'\x00\x00\x00p'
transaction_data_format = b'\x00\x00\x00|'
transaction_id = b'\x00\x00\x00a'
transaction_id_list = b'\x00\x00\x00b'
transaction_tree = b'\x00\x00\x00r'
transactions = b'\x00\x00\x00q'
txid_having_cross_ref = b'\x00\x00\x00z'
until = b'\x00\x00\x00j'
user_id = b'\x00\x00\x00`'
user_list = b'\x00\x00\x009'

```

```
class bbcl.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 bbcl.core.message_key_types.PayloadType
```

```
    Bases: object
```

```
    Type_any = 1
```

```
    Type_binary = 0
```

```
    Type_encrypted_msgpack = 3
```

```
    Type_msgpack = 2
```

```
bbcl.core.message_key_types.convert_from_binary(data_type, dat)
```

```
    Deserialization from simple serialization
```

`bbc1.core.message_key_types.derive_shared_key` (*private\_key*, *shared\_info*, *serialized\_pubkey*)  
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** (*obj*) – callback method that will be called periodically
- **callback\_error** (*obj*) – 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.

**class** `bbc1.core.repair_manager.RepairManager` (*network=None, domain\_id=None, workingdir='.', loglevel='all', logname=None*)

Bases: `object`

Data repair manager for forged transaction/asset

**REQUEST\_REPAIR\_ASSET\_FILE** = 1

**REQUEST\_REPAIR\_TRANSACTION** = 0

**REQUEST\_TO\_SEND\_ASSET\_FILE** = 4

**REQUEST\_TO\_SEND\_TRANSACTION\_DATA** = 2

**RESPONSE\_ASSET\_FILE** = 5

**RESPONSE\_TRANSACTION\_DATA** = 3

**exit\_loop** ()  
Exit the manager loop

**put\_message** (*msg=None*)  
append a message to the queue

### 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.

```
class bbc1.core.topology_manager.TopologyManagerBase (network=None, config=None, domain_id=None, node_id=None, loglevel='all', logname=None)
```

Bases: object

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'\x00\x00'

**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 bbcl.core.user_message_routing.UserMessageRouting (networking, domain_id,
                                                    loglevel='all', log-
                                                    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'\x00\x02'
```

```
RESPONSE_USER_LOCATION = b'\x00\x01'
```

```
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

```
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 `message_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)`

### 1.1.1.2 Module contents

## 1.2 Module contents



## CHAPTER 2

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`



**b**

bbc1, 37  
bbc1.core, 37  
bbc1.core.bbc\_app, 1  
bbc1.core.bbc\_config, 12  
bbc1.core.bbc\_core, 13  
bbc1.core.bbc\_error, 15  
bbc1.core.bbc\_network, 15  
bbc1.core.bbc\_stats, 18  
bbc1.core.bbclib, 19  
bbc1.core.command, 19  
bbc1.core.data\_handler, 19  
bbc1.core.domain0\_manager, 26  
bbc1.core.key\_exchange\_manager, 27  
bbc1.core.logger, 28  
bbc1.core.message\_key\_types, 29  
bbc1.core.query\_management, 32  
bbc1.core.repair\_manager, 33  
bbc1.core.topology\_manager, 34  
bbc1.core.user\_message\_routing, 34



## A

activate\_ledgersubsystem() (in module *bbc1.core.bbc\_core*), 14  
 add() (*bbc1.core.bbc\_network.NeighborInfo* method), 17  
 add\_neighbor() (*bbc1.core.bbc\_network.BBcNetwork* method), 15  
 admin\_info (*bbc1.core.message\_key\_types.KeyType* attribute), 29  
 ADV\_DOMAIN\_LIST (*bbc1.core.domain0\_manager.Domain0Manager* attribute), 26  
 all\_asset\_files (*bbc1.core.message\_key\_types.KeyType* attribute), 29  
 all\_included (*bbc1.core.message\_key\_types.KeyType* attribute), 29  
 anycast\_ttl (*bbc1.core.message\_key\_types.KeyType* attribute), 29  
 asset\_file (*bbc1.core.message\_key\_types.KeyType* attribute), 29  
 asset\_group\_id (*bbc1.core.message\_key\_types.KeyType* attribute), 29  
 asset\_group\_ids (*bbc1.core.message\_key\_types.KeyType* attribute), 29  
 asset\_id (*bbc1.core.message\_key\_types.KeyType* attribute), 29

## B

*bbc1* (module), 37  
*bbc1.core* (module), 37  
*bbc1.core.bbc\_app* (module), 1  
*bbc1.core.bbc\_config* (module), 12  
*bbc1.core.bbc\_core* (module), 13  
*bbc1.core.bbc\_error* (module), 15  
*bbc1.core.bbc\_network* (module), 15  
*bbc1.core.bbc\_stats* (module), 18  
*bbc1.core.bbclib* (module), 19  
*bbc1.core.command* (module), 19  
*bbc1.core.data\_handler* (module), 19  
*bbc1.core.domain0\_manager* (module), 26

*bbc1.core.key\_exchange\_manager* (module), 27  
*bbc1.core.logger* (module), 28  
*bbc1.core.message\_key\_types* (module), 29  
*bbc1.core.query\_management* (module), 32  
*bbc1.core.repair\_manager* (module), 33  
*bbc1.core.topology\_manager* (module), 34  
*bbc1.core.user\_message\_routing* (module), 34  
 bbm configuration  
*bbc1.core.message\_key\_types.KeyType* attribute), 29  
*BBcAppClient* (class in *bbc1.core.bbc\_app*), 1  
*BBcConfig* (class in *bbc1.core.bbc\_config*), 12  
*BBcCoreService* (class in *bbc1.core.bbc\_core*), 13  
*BBcNetwork* (class in *bbc1.core.bbc\_network*), 15  
*BBcStats* (class in *bbc1.core.bbc\_stats*), 18  
 broadcast\_message\_in\_network() (*bbc1.core.bbc\_network.BBcNetwork* method), 15

## C

Callback (class in *bbc1.core.bbc\_app*), 8  
 callback() (*bbc1.core.query\_management.QueryEntry* method), 32  
 callback\_error() (*bbc1.core.query\_management.QueryEntry* method), 32  
 cancel\_insert\_completion\_notification() (*bbc1.core.bbc\_app.BBcAppClient* method), 1  
 CATEGORY\_DATA (*bbc1.core.message\_key\_types.InfraMessageCategory* attribute), 29  
 CATEGORY\_DOMAIN0 (*bbc1.core.message\_key\_types.InfraMessageCategory* attribute), 29  
 CATEGORY\_NETWORK (*bbc1.core.message\_key\_types.InfraMessageCategory* attribute), 29  
 CATEGORY\_TOPOLOGY (*bbc1.core.message\_key\_types.InfraMessageCategory* attribute), 29  
 CATEGORY\_USER (*bbc1.core.message\_key\_types.InfraMessageCategory* attribute), 29

check\_admin\_signature() (*bbc1.core.bbc\_network.BBcNetwork* method), 16  
 check\_table\_existence() (*bbc1.core.data\_handler.DbAdaptor* method), 25  
 check\_table\_existence() (*bbc1.core.data\_handler.MysqlAdaptor* method), 25  
 check\_table\_existence() (*bbc1.core.data\_handler.SqliteAdaptor* method), 26  
 clear\_stats() (*bbc1.core.bbc\_stats.BBcStats* method), 19  
 command (*bbc1.core.message\_key\_types.KeyType* attribute), 29  
 compromised\_asset\_files (*bbc1.core.message\_key\_types.KeyType* attribute), 29  
 compromised\_transaction\_data (*bbc1.core.message\_key\_types.KeyType* attribute), 29  
 compromised\_transaction\_ids (*bbc1.core.message\_key\_types.KeyType* attribute), 29  
 compromised\_transactions (*bbc1.core.message\_key\_types.KeyType* attribute), 29  
 CONFIRM\_KEY\_EXCHANGE (*bbc1.core.bbc\_network.BBcNetwork* attribute), 15  
 convert\_from\_binary() (in module *bbc1.core.message\_key\_types*), 31  
 count (*bbc1.core.message\_key\_types.KeyType* attribute), 29  
 count\_domain\_in\_cross\_ref() (*bbc1.core.data\_handler.DataHandler* method), 20  
 count\_transactions() (*bbc1.core.bbc\_app.BBcAppClient* method), 1  
 count\_transactions() (*bbc1.core.bbc\_core.BBcCoreService* method), 13  
 count\_transactions() (*bbc1.core.data\_handler.DataHandler* method), 20  
 create\_domain() (*bbc1.core.bbc\_network.BBcNetwork* method), 16  
 create\_queue() (*bbc1.core.bbc\_app.Callback* method), 8  
 create\_table() (*bbc1.core.data\_handler.DbAdaptor* method), 25  
 create\_table() (*bbc1.core.data\_handler.MysqlAdaptor* method), 25  
 create\_table() (*bbc1.core.data\_handler.SqliteAdaptor* method), 26  
 create\_ver2\_column() (*bbc1.core.data\_handler.DbAdaptor* method), 25  
 cross\_ref (*bbc1.core.message\_key\_types.KeyType* attribute), 29  
 CROSS\_REF\_ASSIGNMENT (*bbc1.core.user\_message\_routing.UserMessageRouting* attribute), 35  
 CROSS\_REF\_PROBABILITY (*bbc1.core.domain0\_manager.Domain0Manager* attribute), 26  
 cross\_ref\_registered() (*bbc1.core.domain0\_manager.Domain0Manager* method), 27  
 cross\_ref\_verification\_info (*bbc1.core.message\_key\_types.KeyType* attribute), 29  
**D**  
 daemonize() (in module *bbc1.core.bbc\_core*), 14  
 DataHandler (class in *bbc1.core.data\_handler*), 19  
 DataHandlerDomain0 (class in *bbc1.core.data\_handler*), 23  
 DbAdaptor (class in *bbc1.core.data\_handler*), 25  
 deactivate() (*bbc1.core.query\_management.QueryEntry* method), 32  
 del\_entry() (*bbc1.core.query\_management.Ticker* method), 33  
 derive\_shared\_key() (in module *bbc1.core.message\_key\_types*), 31  
 deserialize\_data() (in module *bbc1.core.message\_key\_types*), 32  
 destination\_node\_id (*bbc1.core.message\_key\_types.KeyType* attribute), 29  
 destination\_user\_id (*bbc1.core.message\_key\_types.KeyType* attribute), 29  
 destination\_user\_ids (*bbc1.core.message\_key\_types.KeyType* attribute), 30  
 direct\_send\_to\_user() (in module *bbc1.core.user\_message\_routing*), 36  
 direction (*bbc1.core.message\_key\_types.KeyType* attribute), 30  
 dispatch() (*bbc1.core.bbc\_app.Callback* method), 8  
 DISTRIBUTE\_CROSS\_REF (*bbc1.core.domain0\_manager.Domain0Manager* attribute), 26  
 distribute\_cross\_ref\_in\_domain0() (*bbc1.core.domain0\_manager.Domain0Manager* method), 27



Domain0Manager (class in `bbc1.core.domain0_manager`), 26

DOMAIN\_ACCEPTANCE\_RECOVER\_INTERVAL (bbc1.core.domain0\_manager.Domain0Manager attribute), 26

domain\_close() (bbc1.core.bbc\_app.BBcAppClient method), 2

domain\_id (bbc1.core.message\_key\_types.KeyType attribute), 30

DOMAIN\_INFO\_ADVERTISE\_INTERVAL (bbc1.core.domain0\_manager.Domain0Manager attribute), 26

DOMAIN\_INFO\_LIFETIME (bbc1.core.domain0\_manager.Domain0Manager attribute), 26

domain\_list (bbc1.core.message\_key\_types.KeyType attribute), 30

domain\_ping (bbc1.core.message\_key\_types.KeyType attribute), 30

domain\_setup() (bbc1.core.bbc\_app.BBcAppClient method), 2

**E**

ecdh (bbc1.core.message\_key\_types.KeyType attribute), 30

exchange\_key() (bbc1.core.bbc\_app.BBcAppClient method), 2

exec\_sql() (bbc1.core.data\_handler.DataHandler method), 20

exec\_sql() (bbc1.core.data\_handler.DataHandlerDomain0 method), 23

exit\_loop() (bbc1.core.repair\_manager.RepairManager method), 34

external\_ip4addr (bbc1.core.message\_key\_types.KeyType attribute), 30

external\_ip6addr (bbc1.core.message\_key\_types.KeyType attribute), 30

**F**

forwarding\_list (bbc1.core.message\_key\_types.KeyType attribute), 30

**G**

gather\_signatures() (bbc1.core.bbc\_app.BBcAppClient method), 2

get\_asset\_info() (bbc1.core.data\_handler.DataHandler method), 21

get\_asset\_info() (bbc1.core.data\_handler.DataHandlerDomain0 method), 23

get\_bbc\_config() (bbc1.core.bbc\_app.BBcAppClient method), 3

get\_config() (bbc1.core.bbc\_config.BBcConfig method), 12

get\_domain\_config() (bbc1.core.bbc\_config.BBcConfig method), 12

get\_domain\_keypair() (bbc1.core.bbc\_network.BBcNetwork method), 16

get\_domain\_list() (bbc1.core.bbc\_app.BBcAppClient method), 3

get\_domain\_neighborlist() (bbc1.core.bbc\_app.BBcAppClient method), 3

get\_ECDH\_parameters() (in module `bbc1.core.message_key_types`), 32

get\_entry() (bbc1.core.query\_management.Ticker method), 33

get\_forwarding\_list() (bbc1.core.bbc\_app.BBcAppClient method), 3

get\_from\_queue() (bbc1.core.bbc\_app.Callback method), 8

get\_in\_storage() (bbc1.core.data\_handler.DataHandler method), 21

get\_in\_storage() (bbc1.core.data\_handler.DataHandlerDomain0 method), 23

get\_json\_config() (bbc1.core.bbc\_config.BBcConfig method), 12

get\_logger() (in module `bbc1.core.logger`), 28

get\_node\_id() (bbc1.core.bbc\_app.BBcAppClient method), 3

get\_nodeinfo() (bbc1.core.bbc\_network.NodeInfo method), 18

get\_notification\_list() (bbc1.core.bbc\_app.BBcAppClient method), 3

get\_stats() (bbc1.core.bbc\_app.BBcAppClient method), 3

get\_stats() (bbc1.core.bbc\_stats.BBcStats method), 19

get\_ticker() (in module `bbc1.core.query_management`), 33

get\_user\_list() (bbc1.core.bbc\_app.BBcAppClient method), 3

get\_version() (bbc1.core.data\_handler.DbAdaptor method), 25

**H**

HEADER\_LEN (bbc1.core.message\_key\_types.Message attribute), 31

hint (bbc1.core.message\_key\_types.KeyType attribute), 30

hop\_count (bbc1.core.message\_key\_types.KeyType attribute), 30

**I**

include\_admin\_info() (bbc1.core.bbc\_app.BBcAppClient method), 3

include\_admin\_info\_into\_message\_if\_needed(*bbc1.core.bbc\_network.BBcNetwork* method), 16

include\_cross\_ref(*bbc1.core.bbc\_app.BBcAppClient* method), 3

infra\_command(*bbc1.core.message\_key\_types.KeyType* attribute), 30

infra\_msg\_type(*bbc1.core.message\_key\_types.KeyType* attribute), 30

InfraMessageCategory (class in *bbc1.core.message\_key\_types*), 29

INITIAL\_ACCEPT\_LIMIT (*bbc1.core.domain0\_manager.Domain0Manager* attribute), 27

insert\_cross\_ref(*bbc1.core.data\_handler.DataHandler* method), 21

insert\_transaction(*bbc1.core.bbc\_app.BBcAppClient* method), 4

insert\_transaction(*bbc1.core.bbc\_core.BBcCoreService* method), 13

insert\_transaction(*bbc1.core.data\_handler.DataHandler* method), 21

insert\_transaction(*bbc1.core.data\_handler.DataHandlerDomain0* method), 23

ipv4\_address(*bbc1.core.message\_key\_types.KeyType* attribute), 30

ipv6\_address(*bbc1.core.message\_key\_types.KeyType* attribute), 30

is\_anycast (*bbc1.core.message\_key\_types.KeyType* attribute), 30

is\_less\_than() (in module *bbc1.core.bbc\_network*), 18

is\_replication(*bbc1.core.message\_key\_types.KeyType* attribute), 30

**J**

JOIN\_MULTICAST\_RECEIVER (*bbc1.core.user\_message\_routing.UserMessageRouting* attribute), 35

**K**

KEY\_EXCHANGE\_INVOKE\_MAX\_BACKOFF (*bbc1.core.key\_exchange\_manager.KeyExchangeManager* attribute), 27

KEY\_EXCHANGE\_RETRY\_INTERVAL (*bbc1.core.key\_exchange\_manager.KeyExchangeManager* attribute), 28

KEY\_OBSOLETE\_TIMER (*bbc1.core.key\_exchange\_manager.KeyExchangeManager* attribute), 28

KEY\_REFRESH\_INTERVAL (*bbc1.core.key\_exchange\_manager.KeyExchangeManager* attribute), 28

KeyExchangeManager (class in *bbc1.core.key\_exchange\_manager*), 27

KeyType (class in *bbc1.core.message\_key\_types*), 29

LEAVE\_MULTICAST\_RECEIVER (*bbc1.core.user\_message\_routing.UserMessageRouting* attribute), 35

ledger\_subsys\_manip (*bbc1.core.message\_key\_types.KeyType* attribute), 30

ledger\_subsys\_register (*bbc1.core.message\_key\_types.KeyType* attribute), 30

ledger\_subsys\_verify (*bbc1.core.message\_key\_types.KeyType* attribute), 30

load\_config() (in module *bbc1.core.bbc\_config*), 12

**M**

make\_binary() (in module *bbc1.core.message\_key\_types*), 32

make\_dictionary\_from\_TLV\_format() (in module *bbc1.core.message\_key\_types*), 32

make\_message() (in module *bbc1.core.message\_key\_types*), 32

make\_neighbor\_list() (*bbc1.core.topology\_manager.TopologyManagerBase* method), 34

make\_TLV\_formatted\_message() (in module *bbc1.core.message\_key\_types*), 32

manipulate\_ledger\_subsystem(*bbc1.core.bbc\_app.BBcAppClient* method), 4

MAX\_CROSS\_REF\_STOCK (*bbc1.core.user\_message\_routing.UserMessageRouting* attribute), 35

merkle\_tree (*bbc1.core.message\_key\_types.KeyType* attribute), 30

message (*bbc1.core.message\_key\_types.KeyType* attribute), 30

Message (class in *bbc1.core.message\_key\_types*), 31

message\_seq (*bbc1.core.message\_key\_types.KeyType* attribute), 30

MessageAdaptor (class in *bbc1.core.data\_handler*), 25

**N**

neighbor\_list (*bbc1.core.message\_key\_types.KeyType* attribute), 30

NEIGHBOR\_LIST\_REFRESH\_INTERVAL (*bbc1.core.topology\_manager.TopologyManagerBase* attribute), 34

NeighborInfo (class in *bbc1.core.bbc\_network*), 17  
node\_id (*bbc1.core.message\_key\_types.KeyType* attribute), 30  
node\_info (*bbc1.core.message\_key\_types.KeyType* attribute), 30  
NodeInfo (class in *bbc1.core.bbc\_network*), 18  
NODEINFO\_LIFETIME (*bbc1.core.bbc\_network.NeighborInfo* attribute), 17  
nodekey\_signature (*bbc1.core.message\_key\_types.KeyType* attribute), 30  
nonce (*bbc1.core.message\_key\_types.KeyType* attribute), 30  
notification\_list (*bbc1.core.message\_key\_types.KeyType* attribute), 30  
NOTIFY\_CROSS\_REF\_REGISTERED (*bbc1.core.domain0\_manager.Domain0Manager* attribute), 27  
notify\_domain\_key\_update () (*bbc1.core.bbc\_app.BBcAppClient* method), 4  
NOTIFY\_INSERTED (*bbc1.core.data\_handler.DataHandler* attribute), 20  
NOTIFY\_LEAVE (*bbc1.core.bbc\_network.BBcNetwork* attribute), 15  
NOTIFY\_NEIGHBOR\_LIST (*bbc1.core.topology\_manager.TopologyManagerBase* attribute), 34  
notify\_neighbor\_update () (*bbc1.core.topology\_manager.TopologyManagerBase* method), 34  
NUM\_OF\_COPIES (*bbc1.core.domain0\_manager.Domain0Manager* attribute), 27

**O**

on\_multinodes (*bbc1.core.message\_key\_types.KeyType* attribute), 30  
open\_db () (*bbc1.core.data\_handler.DbAdaptor* method), 25  
open\_db () (*bbc1.core.data\_handler.MysqlAdaptor* method), 26  
open\_db () (*bbc1.core.data\_handler.SqliteAdaptor* method), 26  
outer\_domain\_id (*bbc1.core.message\_key\_types.KeyType* attribute), 30

**P**

parse () (*bbc1.core.message\_key\_types.Message* method), 31  
parser () (in module *bbc1.core.command*), 19  
PayloadType (class in *bbc1.core.message\_key\_types*), 31  
port\_number (*bbc1.core.message\_key\_types.KeyType* attribute), 30  
proc\_cmd\_sign\_request () (*bbc1.core.bbc\_app.Callback* method), 8  
proc\_notify\_cross\_ref () (*bbc1.core.bbc\_app.Callback* method), 8  
proc\_notify\_inserted () (*bbc1.core.bbc\_app.Callback* method), 9  
proc\_resp\_count\_transactions () (*bbc1.core.bbc\_app.Callback* method), 9  
proc\_resp\_cross\_ref\_list () (*bbc1.core.bbc\_app.Callback* method), 9  
proc\_resp\_domain\_close () (*bbc1.core.bbc\_app.Callback* method), 9  
proc\_resp\_domain\_setup () (*bbc1.core.bbc\_app.Callback* method), 9  
proc\_resp\_ecdh\_key\_exchange () (*bbc1.core.bbc\_app.Callback* method), 9  
proc\_resp\_gather\_signature () (*bbc1.core.bbc\_app.Callback* method), 9  
proc\_resp\_get\_config () (*bbc1.core.bbc\_app.Callback* method), 9  
proc\_resp\_get\_domainlist () (*bbc1.core.bbc\_app.Callback* method), 9  
proc\_resp\_get\_forwardinglist () (*bbc1.core.bbc\_app.Callback* method), 10  
proc\_resp\_get\_neighborlist () (*bbc1.core.bbc\_app.Callback* method), 10  
proc\_resp\_get\_node\_id () (*bbc1.core.bbc\_app.Callback* method), 10  
proc\_resp\_get\_notificationlist () (*bbc1.core.bbc\_app.Callback* method), 10  
proc\_resp\_get\_stats () (*bbc1.core.bbc\_app.Callback* method), 10  
proc\_resp\_get\_userlist () (*bbc1.core.bbc\_app.Callback* method), 10  
proc\_resp\_insert () (*bbc1.core.bbc\_app.Callback* method), 10  
proc\_resp\_ledger\_subsystem () (*bbc1.core.bbc\_app.Callback* method), 10  
proc\_resp\_register\_hash () (*bbc1.core.bbc\_app.Callback* method), 10  
proc\_resp\_search\_transaction () (*bbc1.core.bbc\_app.Callback* method), 11  
proc\_resp\_search\_with\_condition () (*bbc1.core.bbc\_app.Callback* method), 11  
proc\_resp\_set\_neighbor () (*bbc1.core.bbc\_app.Callback* method), 11  
proc\_resp\_sign\_request () (*bbc1.core.bbc\_app.Callback* method), 11  
proc\_resp\_traverse\_transactions () (*bbc1.core.bbc\_app.Callback* method), 11  
proc\_resp\_verify\_cross\_ref () (*bbc1.core.bbc\_app.Callback* method), 11

proc\_resp\_verify\_hash() (bbc1.core.bbc\_app.Callback method), 11

proc\_user\_message() (bbc1.core.bbc\_app.Callback method), 11

process\_message() (bbc1.core.data\_handler.DataHandler method), 21

process\_message() (bbc1.core.data\_handler.DataHandlerDomain0 method), 24

process\_message() (bbc1.core.domain0\_manager.Domain0Manager method), 27

process\_message() (bbc1.core.topology\_manager.TopologyManagerBase method), 34

process\_message() (bbc1.core.user\_message\_routing.UserMessageRouting method), 35

process\_message() (bbc1.core.user\_message\_routing.UserMessageRoutingDummy method), 36

purge() (bbc1.core.bbc\_network.NeighborInfo method), 17

PURGE\_INTERVAL\_SEC (bbc1.core.bbc\_network.NeighborInfo attribute), 17

put\_message() (bbc1.core.repair\_manager.RepairManager method), 34

## Q

query\_id (bbc1.core.message\_key\_types.KeyType attribute), 30

QueryEntry (class in bbc1.core.query\_management), 32

quit\_program() (bbc1.core.bbc\_core.BBcCoreService method), 13

## R

random (bbc1.core.message\_key\_types.KeyType attribute), 30

read\_config() (bbc1.core.bbc\_config.BBcConfig method), 12

reason (bbc1.core.message\_key\_types.KeyType attribute), 30

receive\_confirmation() (bbc1.core.key\_exchange\_manager.KeyExchangeManager method), 28

receive\_exchange\_request() (bbc1.core.key\_exchange\_manager.KeyExchangeManager method), 28

receive\_exchange\_response() (bbc1.core.key\_exchange\_manager.KeyExchangeManager method), 28

receiver\_loop() (bbc1.core.bbc\_app.BBcAppClient method), 4

recv() (bbc1.core.message\_key\_types.Message method), 31

ref\_index (bbc1.core.message\_key\_types.KeyType attribute), 30

REFRESH\_FORWARDING\_LIST\_INTERVAL (bbc1.core.user\_message\_routing.UserMessageRouting attribute), 35

register\_in\_ledger\_subsystem() (bbc1.core.bbc\_app.BBcAppClient method), 4

register\_to\_core() (bbc1.core.bbc\_app.BBcAppClient method), 4

register\_user() (bbc1.core.user\_message\_routing.UserMessageRouting method), 35

register\_user() (bbc1.core.user\_message\_routing.UserMessageRouting method), 36

remove() (bbc1.core.bbc\_network.NeighborInfo method), 18

remove() (bbc1.core.data\_handler.DataHandler method), 21

remove() (bbc1.core.data\_handler.DataHandlerDomain0 method), 24

remove\_domain() (bbc1.core.bbc\_network.BBcNetwork method), 16

remove\_domain\_config() (bbc1.core.bbc\_config.BBcConfig method), 12

remove\_from\_notification\_list() (bbc1.core.bbc\_core.BBcCoreService method), 13

remove\_old\_key() (in module bbc1.core.key\_exchange\_manager), 28

remove\_stat\_category() (bbc1.core.bbc\_stats.BBcStats method), 19

remove\_stat\_item() (bbc1.core.bbc\_stats.BBcStats method), 19

REPAIR\_TRANSACTION\_DATA (bbc1.core.data\_handler.DataHandler attribute), 20

RepairManager (class in bbc1.core.repair\_manager), 33

REPLICATION\_ALL (bbc1.core.data\_handler.DataHandler attribute), 20

REPLICATION\_CROSS\_REF (bbc1.core.data\_handler.DataHandler attribute), 20

REPLICATION\_EXT (bbc1.core.data\_handler.DataHandler attribute), 20

REPLICATION\_P2P (bbc1.core.data\_handler.DataHandler attribute), 20

request\_cross\_ref\_holders\_list()

(*bbc1.core.bbc\_app.BBcAppClient method*), 4  
 request\_insert\_completion\_notification()  
 (*bbc1.core.bbc\_app.BBcAppClient method*), 5  
 REQUEST\_KEY\_EXCHANGE  
 (*bbc1.core.bbc\_network.BBcNetwork attribute*), 15  
 REQUEST\_REPAIR\_ASSET\_FILE  
 (*bbc1.core.repair\_manager.RepairManager attribute*), 33  
 REQUEST\_REPAIR\_TRANSACTION  
 (*bbc1.core.repair\_manager.RepairManager attribute*), 33  
 REQUEST\_REPLICATION\_INSERT  
 (*bbc1.core.data\_handler.DataHandler attribute*), 20  
 REQUEST\_SEARCH (*bbc1.core.data\_handler.DataHandler attribute*), 20  
 request\_to\_repair\_asset()  
 (*bbc1.core.bbc\_app.BBcAppClient method*), 5  
 request\_to\_repair\_transaction()  
 (*bbc1.core.bbc\_app.BBcAppClient method*), 5  
 REQUEST\_TO\_SEND\_ASSET\_FILE  
 (*bbc1.core.repair\_manager.RepairManager attribute*), 33  
 REQUEST\_TO\_SEND\_TRANSACTION\_DATA  
 (*bbc1.core.repair\_manager.RepairManager attribute*), 33  
 REQUEST\_VERIFY (*bbc1.core.domain0\_manager.Domain0Manager attribute*), 27  
 request\_verify\_by\_cross\_ref()  
 (*bbc1.core.bbc\_app.BBcAppClient method*), 5  
 REQUEST\_VERIFY\_FROM\_OUTER\_DOMAIN  
 (*bbc1.core.domain0\_manager.Domain0Manager attribute*), 27  
 RESOLVE\_TIMEOUT (*bbc1.core.user\_message\_routing.UserMessageRouting attribute*), 35  
 RESOLVE\_USER\_LOCATION  
 (*bbc1.core.user\_message\_routing.UserMessageRouting attribute*), 35  
 RESPONSE\_ASSET\_FILE  
 (*bbc1.core.repair\_manager.RepairManager attribute*), 33  
 RESPONSE\_KEY\_EXCHANGE  
 (*bbc1.core.bbc\_network.BBcNetwork attribute*), 15  
 RESPONSE\_NO\_SUCH\_USER  
 (*bbc1.core.user\_message\_routing.UserMessageRouting attribute*), 35  
 RESPONSE\_REPLICATION\_INSERT  
 (*bbc1.core.data\_handler.DataHandler attribute*), 20  
 RESPONSE\_SEARCH (*bbc1.core.data\_handler.DataHandler attribute*), 20  
 RESPONSE\_TRANSACTION\_DATA  
 (*bbc1.core.repair\_manager.RepairManager attribute*), 33  
 RESPONSE\_USER\_LOCATION  
 (*bbc1.core.user\_message\_routing.UserMessageRouting attribute*), 35  
 RESPONSE\_VERIFY\_FROM\_OUTER\_DOMAIN  
 (*bbc1.core.domain0\_manager.Domain0Manager attribute*), 27  
 restore\_transaction\_data()  
 (*bbc1.core.data\_handler.DataHandler method*), 22  
 result (*bbc1.core.message\_key\_types.KeyType attribute*), 31  
 retry\_timer (*bbc1.core.message\_key\_types.KeyType attribute*), 31

## S

save\_all\_static\_node\_list()  
 (*bbc1.core.bbc\_network.BBcNetwork method*), 16  
 search\_domain\_having\_cross\_ref()  
 (*bbc1.core.data\_handler.DataHandler method*), 22  
 search\_transaction()  
 (*bbc1.core.bbc\_app.BBcAppClient method*), 5  
 search\_transaction()  
 (*bbc1.core.data\_handler.DataHandler method*), 22  
 search\_transaction()  
 (*bbc1.core.data\_handler.DataHandlerDomain0 method*), 24  
 search\_transaction\_topology()  
 (*bbc1.core.data\_handler.DataHandler method*), 22  
 search\_transaction\_topology()  
 (*bbc1.core.data\_handler.DataHandlerDomain0 method*), 24  
 search\_transaction\_with\_condition()  
 (*bbc1.core.bbc\_app.BBcAppClient method*), 5  
 search\_transaction\_with\_condition()  
 (*bbc1.core.bbc\_core.BBcCoreService method*), 14  
 SECURITY\_STATE\_CONFIRMING  
 (*bbc1.core.bbc\_network.NodeInfo attribute*), 18  
 SECURITY\_STATE\_ESTABLISHED  
 (*bbc1.core.bbc\_network.NodeInfo attribute*), 18  
 SECURITY\_STATE\_NONE  
 (*bbc1.core.bbc\_network.NodeInfo attribute*), 18  
 SECURITY\_STATE\_REQUESTING  
 (*bbc1.core.bbc\_network.NodeInfo attribute*), 18

send\_domain\_ping() (*bbc1.core.bbc\_app.BBcAppClient method*), 6  
 send\_domain\_ping() (*bbc1.core.bbc\_network.BBcNetwork method*), 16  
 send\_inserted\_notification() (*bbc1.core.bbc\_core.BBcCoreService method*), 14  
 send\_key\_exchange\_message() (*bbc1.core.bbc\_network.BBcNetwork method*), 17  
 send\_message() (*bbc1.core.bbc\_app.BBcAppClient method*), 6  
 send\_message\_in\_network() (*bbc1.core.bbc\_network.BBcNetwork method*), 17  
 send\_message\_to\_a\_domain0\_manager() (*bbc1.core.bbc\_network.BBcNetwork method*), 17  
 send\_message\_to\_user() (*bbc1.core.user\_message\_routing.UserMessageRouting method*), 35  
 send\_message\_to\_user() (*bbc1.core.user\_message\_routing.UserMessageRoutingDummy method*), 36  
 send\_multicast\_join() (*bbc1.core.user\_message\_routing.UserMessageRouting method*), 35  
 send\_multicast\_join() (*bbc1.core.user\_message\_routing.UserMessageRoutingDummy method*), 36  
 send\_multicast\_leave() (*bbc1.core.user\_message\_routing.UserMessageRouting method*), 35  
 sendback\_denial\_of\_sign() (*bbc1.core.bbc\_app.BBcAppClient method*), 6  
 sendback\_signature() (*bbc1.core.bbc\_app.BBcAppClient method*), 6  
 set\_aes\_name() (*bbc1.core.user\_message\_routing.UserMessageRouting method*), 35  
 set\_callback() (*bbc1.core.bbc\_app.BBcAppClient method*), 7  
 set\_cipher() (*bbc1.core.key\_exchange\_manager.KeyExchangeManager method*), 28  
 set\_cipher() (in module *bbc1.core.message\_key\_types*), 32  
 set\_client() (*bbc1.core.bbc\_app.Callback method*), 11  
 set\_domain\_id() (*bbc1.core.bbc\_app.BBcAppClient method*), 7  
 set\_domain\_static\_node() (*bbc1.core.bbc\_app.BBcAppClient method*), 7  
 set\_invoke\_timer() (*bbc1.core.key\_exchange\_manager.KeyExchangeManager method*), 34  
 set\_keypair() (*bbc1.core.bbc\_app.BBcAppClient method*), 7  
 set\_logger() (*bbc1.core.bbc\_app.Callback method*), 11  
 set\_node\_key() (*bbc1.core.bbc\_app.BBcAppClient method*), 7  
 set\_user\_id() (*bbc1.core.bbc\_app.BBcAppClient method*), 7  
 setup\_tcp\_server() (*bbc1.core.bbc\_network.BBcNetwork method*), 17  
 setup\_udp\_socket() (*bbc1.core.bbc\_network.BBcNetwork method*), 17  
 show\_list() (*bbc1.core.bbc\_network.NeighborInfo method*), 18  
 signature (*bbc1.core.message\_key\_types.KeyType attribute*), 31  
 source\_domain\_id (*bbc1.core.message\_key\_types.KeyType attribute*), 31  
 source\_node\_id (*bbc1.core.message\_key\_types.KeyType attribute*), 31  
 source\_receiver\_id (*bbc1.core.message\_key\_types.KeyType attribute*), 31  
 SQLiteAdaptor (class in *bbc1.core.data\_handler*), 26  
 state\_from (*bbc1.core.message\_key\_types.KeyType attribute*), 31  
 start\_receiver\_loop() (*bbc1.core.bbc\_app.BBcAppClient method*), 8  
 STATE\_CONFIRMING (*bbc1.core.key\_exchange\_manager.KeyExchangeManager attribute*), 28  
 STATE\_ESTABLISHED (*bbc1.core.key\_exchange\_manager.KeyExchangeManager attribute*), 28  
 STATE\_NONE (*bbc1.core.key\_exchange\_manager.KeyExchangeManager attribute*), 28  
 STATE\_REQUESTING (*bbc1.core.key\_exchange\_manager.KeyExchangeManager attribute*), 28  
 static\_entry (*bbc1.core.message\_key\_types.KeyType attribute*), 31  
 stats (*bbc1.core.message\_key\_types.KeyType attribute*), 31  
 status (*bbc1.core.message\_key\_types.KeyType attribute*), 31  
 stop\_all\_timers() (*bbc1.core.domain0\_manager.Domain0Manager method*), 27  
 stop\_all\_timers() (*bbc1.core.key\_exchange\_manager.KeyExchangeManager method*), 28  
 stop\_all\_timers() (*bbc1.core.topology\_manager.TopologyManagerBase method*), 34

stop\_all\_timers() (bbc1.core.user\_message\_routing.UserMessageRouting attribute), 31  
 method), 36

stop\_all\_timers() (bbc1.core.user\_message\_routing.UserMessageRoutingDummy attribute), 31  
 method), 36

store\_in\_storage() (bbc1.core.data\_handler.DataHandler attribute), 22  
 method), 22

store\_in\_storage() (bbc1.core.data\_handler.DataHandlerDomain0 attribute), 25  
 method), 25

sync\_by\_queryid() (bbc1.core.bbc\_app.Callback attribute), 11  
 method), 11

synchronize() (bbc1.core.bbc\_app.Callback attribute), 12  
 method), 12

## T

tcpserver\_loop() (bbc1.core.bbc\_network.BBcNetwork attribute), 17  
 method), 17

Ticker (class in bbc1.core.query\_management), 33

to\_2byte() (in module bbc1.core.message\_key\_types), 32

to\_4byte() (in module bbc1.core.message\_key\_types), 32

TopologyManagerBase (class in bbc1.core.topology\_manager), 34

touch() (bbc1.core.bbc\_network.NodeInfo attribute), 18  
 method), 18

transaction\_data (bbc1.core.message\_key\_types.KeyType attribute), 31

transaction\_data\_format (bbc1.core.message\_key\_types.KeyType attribute), 31

transaction\_id (bbc1.core.message\_key\_types.KeyType attribute), 31

transaction\_id\_list (bbc1.core.message\_key\_types.KeyType attribute), 31

transaction\_tree (bbc1.core.message\_key\_types.KeyType attribute), 31

transactions (bbc1.core.message\_key\_types.KeyType attribute), 31

traverse\_transactions() (bbc1.core.bbc\_app.BBcAppClient attribute), 8  
 method), 8

txid\_having\_cross\_ref (bbc1.core.message\_key\_types.KeyType attribute), 31

Type\_any (bbc1.core.message\_key\_types.PayloadType attribute), 31

Type\_binary (bbc1.core.message\_key\_types.PayloadType attribute), 31

Type\_encrypted\_msgpack (bbc1.core.message\_key\_types.PayloadType attribute), 31

Type\_msgpack (bbc1.core.message\_key\_types.PayloadType attribute), 31

## U

unregister\_from\_core() (bbc1.core.bbc\_app.BBcAppClient attribute), 8  
 method), 8

unregister\_user() (bbc1.core.user\_message\_routing.UserMessageRouting attribute), 36  
 method), 36

unregister\_user() (bbc1.core.user\_message\_routing.UserMessageRoutingDummy attribute), 36  
 method), 36

unset\_cipher() (bbc1.core.key\_exchange\_manager.KeyExchangeManager attribute), 28  
 method), 28

unset\_cipher() (in module bbc1.core.message\_key\_types), 32

until (bbc1.core.message\_key\_types.KeyType attribute), 31

update() (bbc1.core.bbc\_network.NodeInfo attribute), 18  
 method), 18

update() (bbc1.core.query\_management.QueryEntry attribute), 33  
 method), 33

update\_config() (bbc1.core.bbc\_config.BBcConfig attribute), 12  
 method), 12

update\_deep() (in module bbc1.core.bbc\_config), 12

update\_domain\_belong\_to() (bbc1.core.domain0\_manager.Domain0Manager attribute), 27  
 method), 27

update\_expiration\_time() (bbc1.core.query\_management.QueryEntry attribute), 33  
 method), 33

update\_refresh\_timer\_entry() (bbc1.core.topology\_manager.TopologyManagerBase attribute), 34  
 method), 34

update\_stats() (bbc1.core.bbc\_stats.BBcStats attribute), 19  
 method), 19

update\_stats\_decrement() (bbc1.core.bbc\_stats.BBcStats attribute), 19  
 method), 19

update\_stats\_increment() (bbc1.core.bbc\_stats.BBcStats attribute), 19  
 method), 19

update\_table\_def() (bbc1.core.data\_handler.DbAdaptor attribute), 25  
 method), 25

user\_id (bbc1.core.message\_key\_types.KeyType attribute), 31

user\_list (bbc1.core.message\_key\_types.KeyType attribute), 31

UserMessageRouting (class in bbc1.core.user\_message\_routing), 35

UserMessageRoutingDummy (class in  
*bbc1.core.user\_message\_routing*), 36

## V

validate\_transaction()  
(*bbc1.core.bbc\_core.BBcCoreService* method),  
14

verify\_in\_ledger\_subsystem()  
(*bbc1.core.bbc\_app.BBcAppClient* method), 8