BBc1 Documentation

Release 1.1

beyond-blockchain.org

Nov 14, 2018

Contents:

1 bbc1 package **1**

1.1 Subpackages 1

1.1.1 bbc1.core package 1

1.1.1.1 Subpackages 1

1.1.1.2 Submodules 1

1.1.1.3 Module contents 9

1.2 Module contents 9

2 Indices and tables **11**

Python Module Index **13**

CHAPTER 1

bbc1 package

1.1 Subpackages

1.1.1 bbc1.core package

1.1.1.1 Subpackages

1.1.1.2 Submodules

bbc1.core.bbc_app module

bbc1.core.bbc_config module

bbc1.core.bbc_core module

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

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

bbc1.core.command module

bbc1.core.data_handler module

bbc1.core.domain0_manager module

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\x00\t'`

`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\x00\r'
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'

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'

user_id = b'\x00\x00\x00`'

user_list = b'\x00\x00\x009'

class bbc1.core.message_key_types.**Message**

Bases: object

Message parser

HEADER_LEN = 8

parse()

Parse the message in the buffer

recv(*dat*)

Append message to the buffer

class bbc1.core.message_key_types.**PayloadType**

Bases: object

Type_any = 1

Type_binary = 0

Type_encrypted_msgpack = 3

Type_msgpack = 2

bbc1.core.message_key_types.**convert_from_binary**(*data_type*, *dat*)

Deserialization from simple serialization

bbc1.core.message_key_types.**derive_shared_key**(*private_key*, *serialized_pubkey*, *shared_info*)

Utility for deriving shared key in ECDH procedure

`bbc1.core.message_key_types.`**`deserialize_data`**(*payload_type*, *dat*)
: Utility for deserializing the received message

`bbc1.core.message_key_types.`**`get_ECDH_parameters`**()
: Utility for initialization of ECDH parameters

`bbc1.core.message_key_types.`**`make_TLV_formatted_message`**(*msg*)
: Utility for simple serialization function

`bbc1.core.message_key_types.`**`make_binary`**(*dat*)
: Simple serialize function

  Basically, Type-Length-Value format is created for each item.

`bbc1.core.message_key_types.`**`make_dictionary_from_TLV_format`**(*dat*)
: Utility for simple deserialization function

`bbc1.core.message_key_types.`**`make_message`**(*payload_type*, *msg*, *payload_version=0*, *key_name=None*)
: Utility for making serialized message data

`bbc1.core.message_key_types.`**`set_cipher`**(*shared_key*, *nonce*, *key_name*, *hint*)
: Set shared key to the encryptor and decryptor

  Encryptor and Decryptor are created for each inter-node connection

`bbc1.core.message_key_types.`**`to_2byte`**(*val*, *offset=0*)

`bbc1.core.message_key_types.`**`to_4byte`**(*val*, *offset=0*)

`bbc1.core.message_key_types.`**`unset_cipher`**(*key_name*)

bbc1.core.query_management module

Copyright (c) 2017 beyond-blockchain.org.

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

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

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

`class` `bbc1.core.query_management.`**`QueryEntry`**(*expire_after=30*, *callback_expire=None*, *callback=None*, *callback_error=None*, *interval=0*, *data={}*, *retry_count=-1*)
: Bases: `object`

  Callback manager

  `callback`()
  : Call a callback function for successful case

  `callback_error`()
  : Call a callback function for failure case

  `deactivate`()
  : Deactivate the entry

  `update`(*fire_after=None*, *expire_after=None*, *callback=None*, *callback_error=None*, *init=False*)
  : Update the entry information

Parameters

- **`fire_after`** (*`float`*) – set callback (periodical) to fire after given time (in second)
- **`expire_after`** (*`float`*) – set expiration timer to given time (in second)
- **`callback`** (*`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

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

1.1.1.3 Module contents

1.2 Module contents

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

b

bbc1, 9
bbc1.core, 9
bbc1.core.bbc_error, 1
bbc1.core.bbc_stats, 2
bbc1.core.key_exchange_manager, 2
bbc1.core.logger, 3
bbc1.core.message_key_types, 4
bbc1.core.query_management, 7
bbc1.core.topology_manager, 8

Index

A

admin_info (bbc1.core.message_key_types.KeyType attribute), 4
all_asset_files (bbc1.core.message_key_types.KeyType attribute), 4
all_included (bbc1.core.message_key_types.KeyType attribute), 4
anycast_ttl (bbc1.core.message_key_types.KeyType attribute), 4
asset_file (bbc1.core.message_key_types.KeyType attribute), 4
asset_group_id (bbc1.core.message_key_types.KeyType attribute), 4
asset_group_ids (bbc1.core.message_key_types.KeyType attribute), 4
asset_id (bbc1.core.message_key_types.KeyType attribute), 4

B

bbc1 (module), 9
bbc1.core (module), 9
bbc1.core.bbc_error (module), 1
bbc1.core.bbc_stats (module), 2
bbc1.core.key_exchange_manager (module), 2
bbc1.core.logger (module), 3
bbc1.core.message_key_types (module), 4
bbc1.core.query_management (module), 7
bbc1.core.topology_manager (module), 8
bbc_configuration (bbc1.core.message_key_types.KeyType attribute), 4
BBcStats (class in bbc1.core.bbc_stats), 2

C

callback() (bbc1.core.query_management.QueryEntry method), 7
callback_error() (bbc1.core.query_management.QueryEntry method), 7
CATEGORY_DATA (bbc1.core.message_key_types.InfraMessageCategory attribute), 4
CATEGORY_DOMAIN0 (bbc1.core.message_key_types.InfraMessageCategory attribute), 4
CATEGORY_NETWORK (bbc1.core.message_key_types.InfraMessageCategory attribute), 4
CATEGORY_TOPOLOGY (bbc1.core.message_key_types.InfraMessageCategory attribute), 4
CATEGORY_USER (bbc1.core.message_key_types.InfraMessageCategory attribute), 4
clear_stats() (bbc1.core.bbc_stats.BBcStats method), 2
command (bbc1.core.message_key_types.KeyType attribute), 4
compromised_asset_files (bbc1.core.message_key_types.KeyType attribute), 4
compromised_transaction_data (bbc1.core.message_key_types.KeyType attribute), 4
compromised_transaction_ids (bbc1.core.message_key_types.KeyType attribute), 4
compromised_transactions (bbc1.core.message_key_types.KeyType attribute), 4
convert_from_binary() (in module bbc1.core.message_key_types), 6
count (bbc1.core.message_key_types.KeyType attribute), 4
cross_ref (bbc1.core.message_key_types.KeyType attribute), 4
cross_ref_verification_info (bbc1.core.message_key_types.KeyType attribute), 4

D

deactivate() (bbc1.core.query_management.QueryEntry method), 7

del_entry() (bbc1.core.query_management.Ticker method), 8
derive_shared_key() (in module bbc1.core.message_key_types), 6
deserialize_data() (in module bbc1.core.message_key_types), 6
destination_node_id (bbc1.core.message_key_types.KeyType attribute), 4
destination_user_id (bbc1.core.message_key_types.KeyType attribute), 4
destination_user_ids (bbc1.core.message_key_types.KeyType attribute), 5
direction (bbc1.core.message_key_types.KeyType attribute), 5
domain_id (bbc1.core.message_key_types.KeyType attribute), 5
domain_list (bbc1.core.message_key_types.KeyType attribute), 5
domain_ping (bbc1.core.message_key_types.KeyType attribute), 5

E

ecdh (bbc1.core.message_key_types.KeyType attribute), 5
external_ip4addr (bbc1.core.message_key_types.KeyType attribute), 5
external_ip6addr (bbc1.core.message_key_types.KeyType attribute), 5

F

forwarding_list (bbc1.core.message_key_types.KeyType attribute), 5

G

get_ECDH_parameters() (in module bbc1.core.message_key_types), 7
get_entry() (bbc1.core.query_management.Ticker method), 8
get_logger() (in module bbc1.core.logger), 3
get_stats() (bbc1.core.bbc_stats.BBcStats method), 2
get_ticker() (in module bbc1.core.query_management), 8

H

HEADER_LEN (bbc1.core.message_key_types.Message attribute), 6
hint (bbc1.core.message_key_types.KeyType attribute), 5
hop_count (bbc1.core.message_key_types.KeyType attribute), 5

I

infra_command (bbc1.core.message_key_types.KeyType attribute), 5
infra_msg_type (bbc1.core.message_key_types.KeyType attribute), 5
InfraMessageCategory (class in bbc1.core.message_key_types), 4
ipv4_address (bbc1.core.message_key_types.KeyType attribute), 5
ipv6_address (bbc1.core.message_key_types.KeyType attribute), 5
is_anycast (bbc1.core.message_key_types.KeyType attribute), 5
is_replication (bbc1.core.message_key_types.KeyType attribute), 5

K

KEY_EXCHANGE_INVOKE_MAX_BACKOFF (bbc1.core.key_exchange_manager.KeyExchangeManager attribute), 2
KEY_EXCHANGE_RETRY_INTERVAL (bbc1.core.key_exchange_manager.KeyExchangeManager attribute), 2
KEY_OBSOLETE_TIMER (bbc1.core.key_exchange_manager.KeyExchangeManager attribute), 3
KEY_REFRESH_INTERVAL (bbc1.core.key_exchange_manager.KeyExchangeManager attribute), 3
KeyExchangeManager (class in bbc1.core.key_exchange_manager), 2
KeyType (class in bbc1.core.message_key_types), 4

L

ledger_subsys_manip (bbc1.core.message_key_types.KeyType attribute), 5
ledger_subsys_register (bbc1.core.message_key_types.KeyType attribute), 5
ledger_subsys_verify (bbc1.core.message_key_types.KeyType attribute), 5

M

make_binary() (in module bbc1.core.message_key_types), 7
make_dictionary_from_TLV_format() (in module bbc1.core.message_key_types), 7
make_message() (in module bbc1.core.message_key_types), 7
make_neighbor_list() (bbc1.core.topology_manager.TopologyManagerBase method), 8
make_TLV_formatted_message() (in module bbc1.core.message_key_types), 7
merkle_tree (bbc1.core.message_key_types.KeyType attribute), 5
message (bbc1.core.message_key_types.KeyType attribute), 5
Message (class in bbc1.core.message_key_types), 6
message_seq (bbc1.core.message_key_types.KeyType attribute), 5

N

neighbor_list (bbc1.core.message_key_types.KeyType attribute), 5
NEIGHBOR_LIST_REFRESH_INTERVAL (bbc1.core.topology_manager.TopologyManagerBase attribute), 8
node_id (bbc1.core.message_key_types.KeyType attribute), 5
node_info (bbc1.core.message_key_types.KeyType attribute), 5
nodekey_signature (bbc1.core.message_key_types.KeyType attribute), 5
nonce (bbc1.core.message_key_types.KeyType attribute), 5
notification_list (bbc1.core.message_key_types.KeyType attribute), 5
NOTIFY_NEIGHBOR_LIST (bbc1.core.topology_manager.TopologyManagerBase attribute), 8
notify_neighbor_update() (bbc1.core.topology_manager.TopologyManagerBase method), 8

O

on_multinodes (bbc1.core.message_key_types.KeyType attribute), 5
outer_domain_id (bbc1.core.message_key_types.KeyType attribute), 5

P

parse() (bbc1.core.message_key_types.Message method), 6
PayloadType (class in bbc1.core.message_key_types), 6
port_number (bbc1.core.message_key_types.KeyType attribute), 5
process_message() (bbc1.core.topology_manager.TopologyManagerBase method), 9

Q

query_id (bbc1.core.message_key_types.KeyType attribute), 5
QueryEntry (class in bbc1.core.query_management), 7

R

random (bbc1.core.message_key_types.KeyType attribute), 5
reason (bbc1.core.message_key_types.KeyType attribute), 5
receive_confirmation() (bbc1.core.key_exchange_manager.KeyExchangeManager method), 3
receive_exchange_request() (bbc1.core.key_exchange_manager.KeyExchangeManager method), 3
receive_exchange_response() (bbc1.core.key_exchange_manager.KeyExchangeManager method), 3
recv() (bbc1.core.message_key_types.Message method), 6
ref_index (bbc1.core.message_key_types.KeyType attribute), 5
remove_old_key() (in module bbc1.core.key_exchange_manager), 3
remove_stat_category() (bbc1.core.bbc_stats.BBcStats method), 2
remove_stat_item() (bbc1.core.bbc_stats.BBcStats method), 2
result (bbc1.core.message_key_types.KeyType attribute), 6
retry_timer (bbc1.core.message_key_types.KeyType attribute), 6

S

set_cipher() (bbc1.core.key_exchange_manager.KeyExchangeManager method), 3
set_cipher() (in module bbc1.core.message_key_types), 7
set_invoke_timer() (bbc1.core.key_exchange_manager.KeyExchangeManag method), 3
signature (bbc1.core.message_key_types.KeyType attribute), 6
source_domain_id (bbc1.core.message_key_types.KeyType attribute), 6
source_node_id (bbc1.core.message_key_types.KeyType attribute), 6
source_user_id (bbc1.core.message_key_types.KeyType attribute), 6
STATE_CONFIRMING (bbc1.core.key_exchange_manager.KeyExchangeM attribute), 3
STATE_ESTABLISHED (bbc1.core.key_exchange_manager.KeyExchangeManager attribute), 3
STATE_NONE (bbc1.core.key_exchange_manager.KeyExchangeManager attribute), 3
STATE_REQUESTING (bbc1.core.key_exchange_manager.KeyExchangeM attribute), 3
static_entry (bbc1.core.message_key_types.KeyType attribute), 6
stats (bbc1.core.message_key_types.KeyType attribute), 6
status (bbc1.core.message_key_types.KeyType attribute), 6
stop_all_timers() (bbc1.core.key_exchange_manager.KeyExchangeManager method), 3
stop_all_timers() (bbc1.core.topology_manager.TopologyManagerBase method), 9

T

Ticker (class in bbc1.core.query_management), 8

to_2byte() (in module bbc1.core.message_key_types), 7
to_4byte() (in module bbc1.core.message_key_types), 7
TopologyManagerBase (class in bbc1.core.topology_manager), 8
transaction_data (bbc1.core.message_key_types.KeyType attribute), 6
transaction_data_format (bbc1.core.message_key_types.KeyType attribute), 6
transaction_id (bbc1.core.message_key_types.KeyType attribute), 6
transaction_id_list (bbc1.core.message_key_types.KeyType attribute), 6
transaction_tree (bbc1.core.message_key_types.KeyType attribute), 6
transactions (bbc1.core.message_key_types.KeyType attribute), 6
txid_having_cross_ref (bbc1.core.message_key_types.KeyType attribute), 6
Type_any (bbc1.core.message_key_types.PayloadType attribute), 6
Type_binary (bbc1.core.message_key_types.PayloadType attribute), 6
Type_encrypted_msgpack (bbc1.core.message_key_types.PayloadType attribute), 6
Type_msgpack (bbc1.core.message_key_types.PayloadType attribute), 6

U

unset_cipher() (bbc1.core.key_exchange_manager.KeyExchangeManager method), 3
unset_cipher() (in module bbc1.core.message_key_types), 7
update() (bbc1.core.query_management.QueryEntry method), 7
update_expiration_time() (bbc1.core.query_management.QueryEntry method), 8
update_refresh_timer_entry() (bbc1.core.topology_manager.TopologyManagerBase method), 9
update_stats() (bbc1.core.bbc_stats.BBcStats method), 2
update_stats_decrement() (bbc1.core.bbc_stats.BBcStats method), 2
update_stats_increment() (bbc1.core.bbc_stats.BBcStats method), 2
user_id (bbc1.core.message_key_types.KeyType attribute), 6
user_list (bbc1.core.message_key_types.KeyType attribute), 6