

---

# **py-bbclib Documentation**

***Release 1.4.1***

**beyond-blockchain.org**

**Jul 29, 2019**



---

## Contents:

---

<b>1</b>	<b>bbclib package</b>	<b>1</b>
1.1	Subpackages	1
1.1.1	bbclib.compat package	1
1.1.1.1	Submodules	1
1.1.1.2	Module contents	13
1.1.2	bbclib.libs package	13
1.1.2.1	Submodules	13
1.1.2.2	Module contents	26
1.2	Module contents	26
<b>2</b>	<b>Indices and tables</b>	<b>27</b>
	<b>Python Module Index</b>	<b>29</b>
	<b>Index</b>	<b>31</b>



## 1.1 Subpackages

### 1.1.1 bbclib.compat package

#### 1.1.1.1 Submodules

##### bbclib.compat.bbclib module

Copyright (c) 2019 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** `bbclib.compat.bbclib.BBcAsset` (*user\_id=None, asset\_file=None, asset\_body=None, format\_type=0, id\_length=32*)

Bases: `object`

Asset part in a transaction

**add** (*user\_id=None, asset\_file=None, asset\_body=None*)  
Add parts in this object

**deserialize** (*data*)  
Deserialize into this object

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

**Returns** True if successful

**Return type** bool

**deserialize\_obj** (*obj*)

Deserialize bson/msgpack data into this object

**Parameters** **obj** (*bytes*) – object data

**Returns** True if successful

**Return type** bool

**digest** ()

Calculate the digest

The digest corresponds to the `asset_id` of this object

**Returns** `asset_id` (or digest)

**Return type** bytes

**get\_asset\_file** ()

Get asset file content and its digest

**Returns** digest of the file content bytes: the file content

**Return type** bytes

**get\_dict** (*for\_digest\_calculation=False*)

Serialize this object

**recover\_asset\_file** (*asset\_file, id\_length=32*)

Recover asset file info from the given raw content

**serialize** (*for\_digest\_calculation=False*)

Serialize this object

**Parameters** **for\_digest\_calculation** (*bool*) – True if digest calculation

**Returns** serialized binary data

**Return type** bytes

**class** `bbclib.compat.bbclib.BBcCrossRef` (*domain\_id=None, transaction\_id=None, deserialize=None, format\_type=0*)

Bases: object

CrossRef part in a transaction

**deserialize** (*data*)

Deserialize into this object

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

**Returns** True if successful

**Return type** bool

**deserialize\_obj** (*obj*)

Deserialize bson/msgpack data into this object

**Parameters** **obj** (*bytes*) – object data

**Returns** True if successful

**Return type** bool

**get\_dict** ()

Serialize this object into bson format

**serialize()**

Serialize this object

**Returns** serialized binary data

**Return type** bytes

**class** bbclib.compat.bbclib.**BBcEvent** (*asset\_group\_id=None, format\_type=0, id\_length=32*)

Bases: object

Event part in a transaction

**add** (*asset\_group\_id=None, reference\_index=None, mandatory\_approver=None, op-  
tion\_approver\_num\_numerator=0, option\_approver\_num\_denominator=0, op-  
tion\_approver=None, asset=None*)  
Add parts

**deserialize** (*data*)

Deserialize into this object

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

**Returns** True if successful

**Return type** bool

**deserialize\_obj** (*obj*)

Deserialize bson/msgpack data into this object

**Parameters** *obj* (*bytes*) – object data

**Returns** True if successful

**Return type** bool

**get\_dict** ()

Serialize this object

**serialize** ()

Serialize this object

**Returns** serialized binary data

**Return type** bytes

**class** bbclib.compat.bbclib.**BBcFormat**

Bases: object

**FORMAT\_BINARY** = 0

**FORMAT\_BSON** = 1

**FORMAT\_BSON\_COMPRESS\_BZ2** = 2

**FORMAT\_BSON\_COMPRESS\_ZLIB** = 3

**FORMAT\_MSGPACK** = 4

**FORMAT\_MSGPACK\_COMPRESS\_BZ2** = 5

**FORMAT\_MSGPACK\_COMPRESS\_ZLIB** = 6

**class** bbclib.compat.bbclib.**BBcPointer** (*transaction\_id=None, asset\_id=None, for-  
mat\_type=0, id\_length=32*)

Bases: object

Pointer part in a transaction

**add** (*transaction\_id=None, asset\_id=None*)

Add parts

**deserialize** (*data*)

Deserialize into this object

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

**Returns** True if successful

**Return type** bool

**deserialize\_obj** (*obj*)

Deserialize bson/msgpack data into this object

**Parameters** **obj** (*bytes*) – object data

**Returns** True if successful

**Return type** bool

**get\_dict** ()

Serialize this object

**serialize** ()

Serialize this object

**Returns** serialized binary data

**Return type** bytes

```
class bbclib.compat.bbclib.BBcReference (asset_group_id, transaction,  
                                         ref_transaction=None, event_index_in_ref=0,  
                                         format_type=0, id_length=32)
```

Bases: object

Reference part in a transaction

**add\_signature** (*user\_id=None, signature=None*)

Add signature in the reserved space

**Parameters**

- **user\_id** (*bytes*) – user\_id of the signature owner
- **signature** ([BBcSignature](#)) – signature

**deserialize** (*data*)

Deserialize into this object

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

**Returns** True if successful

**Return type** bool

**deserialize\_obj** (*obj*)

Deserialize bson/msgpack data into this object

**Parameters** **obj** (*bytes*) – object data

**Returns** True if successful

**Return type** bool

**get\_destinations** ()

Return the list of approvers in the referred transaction



**get\_dict()**

Serialize this object

**get\_referred\_transaction()**

Return referred transaction in serialized format

**prepare\_reference(ref\_transaction)**

Read the previous referencing transaction

**serialize()**

Serialize this object

**Returns** serialized binary data

**Return type** bytes

**class** bbclib.compat.bbclib.BBcRelation(asset\_group\_id=None, id\_length=32, format\_type=0)

Bases: object

Relation part in a transaction

**add**(asset\_group\_id=None, asset=None, pointer=None)

Add parts

**deserialize(data)**

Deserialize bson data into this object

**Parameters** **data** (*dict*) – bson data

**Returns** True if successful

**Return type** bool

**deserialize\_obj(obj)**

Deserialize bson/msgpack data into this object

**Parameters** **data** (*bytes*) – object data

**Returns** True if successful

**Return type** bool

**get\_dict()**

Serialize this object

**serialize()**

Serialize this object

**Returns** serialized binary data

**Return type** bytes

**class** bbclib.compat.bbclib.BBcSignature(key\_type=2, deserialize=None, format\_type=0)

Bases: object

Signature part in a transaction

**add**(signature=None, pubkey=None)

Add signature and public key

**deserialize(data)**

Deserialize into this object

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

**Returns** True if successful

**Return type** bool

**deserialize\_obj** (*obj*)

Deserialize bson/msgpack data into this object

**Parameters** **obj** (*bytes*) – object data

**Returns** True if successful

**Return type** bool

**get\_dict** ()

Serialize this object

**serialize** ()

Serialize this object

**verify** (*digest*)

Verify digest using pubkey in signature

**Parameters** **digest** (*bytes*) – digest to verify

**Returns** 0:invalid, 1:valid

**Return type** int

**class** bbclib.compat.bbclib.BBcTransaction (*version=1, deserialize=None, format\_type=0, id\_length=32*)

Bases: object

Transaction object

**WITH\_WIRE** = True

**add** (*event=None, reference=None, relation=None, witness=None, cross\_ref=None*)

Add parts

**add\_signature** (*user\_id=None, signature=None*)

Add signature in the reserved space

**Parameters**

- **user\_id** (*bytes*) – user\_id of the signature owner
- **signature** (BBcSignature) – signature

**Returns** True if successful

**Return type** bool

**deserialize** (*data*)

Deserialize into this object

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

**Returns** True if successful

**Return type** bool

**deserialize\_obj** (*data*)

Deserialize bson/msgpack data into this object

**Parameters** **data** (*bytes*) – object data

**Returns** True if successful

**Return type** bool

**digest** ()

Calculate the digest

The digest corresponds to the transaction\_id of this object

**Returns** transaction\_id (or digest)

**Return type** bytes

**get\_sig\_index** (user\_id)

Reserve a space for signature for the specified user\_id

**Parameters** **user\_id** (bytes) – user\_id whose signature will be added to the signature part

**Returns** position (index) in the signature part

**Return type** int

**serialize** (for\_id=False)

Serialize the whole parts

**serialize\_obj** (for\_id=False, no\_header=False)

Serialize the whole parts

**set\_format\_type** (format\_type)

**sign** (key\_type=2, private\_key=None, public\_key=None, keypair=None)

Sign the transaction

**Parameters**

- **key\_type** (int) – Type of encryption key's curve
- **private\_key** (bytes) –
- **public\_key** (bytes) –
- **keypair** (KeyPair) – keypair or set of private\_key and public\_key needs to be given

**Returns**

**Return type** BBcSignature

**class** bbclib.compat.bbclib.BBcWitness (format\_type=0, id\_length=32)

Bases: object

Witness part in a transaction

**add\_signature** (user\_id=None, signature=None)

Add signature in the reserved space for the user\_id that was registered before

**Parameters**

- **user\_id** (bytes) – user\_id of the signature owner
- **signature** (bytes) – signature

**add\_witness** (user\_id)

Register user\_id in the list

**deserialize** (data)

Deserialize into this object

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

**Returns** True if successful

**Return type** bool

**deserialize\_obj** (*obj*)

Deserialize bson/msgpack data into this object

**Parameters** **obj** (*bytes*) – object data

**Returns** True if successful

**Return type** bool

**get\_dict** ()

Serialize this object

**serialize** ()

Serialize this object

**Returns** serialized binary data

**Return type** bytes

**class** bbclib.compat.bbclib.**KeyPair** (*curvetype=2, compression=False, privkey=None, pubkey=None*)

Bases: object

**POINT\_CONVERSION\_COMPRESSED** = 2

**POINT\_CONVERSION\_UNCOMPRESSED** = 4

Key pair container

**generate** ()

Generate a new key pair

**get\_private\_key\_in\_der** ()

Return private key in DER format

**get\_private\_key\_in\_pem** ()

Return private key in PEM format

**get\_public\_key\_in\_pem** ()

Return public key in PEM format

**import\_publickey\_cert\_pem** (*cert\_pemstring, privkey\_pemstring=None*)

Verify and import X509 public key certificate in pem format

**mk\_keyobj\_from\_private\_key** ()

Make a keypair object from the binary data of private key

**mk\_keyobj\_from\_private\_key\_der** (*derdat*)

Make a keypair object from the private key in DER format

**mk\_keyobj\_from\_private\_key\_pem** (*pemdat\_string*)

Make a keypair object from the private key in PEM format

**sign** (*digest*)

Sign to the given value

**Parameters** **digest** (*bytes*) – given value

**Returns** signature

**Return type** bytes

**to\_binary** (*dat*)

**verify** (*digest, sig*)

Verify the digest and the signature using the private key in this object

```
class bbclib.compat.bbclib.KeyType
    Bases: object

    ECDSA_P256v1 = 2
    ECDSA_SECP256k1 = 1
    NOT_INITIALIZED = 0

class bbclib.compat.bbclib.MsgType
    Bases: object

    Message types for between core node and client

    CANCEL_INSERT_NOTIFICATION = 16
    DOMAIN_PING = 12
    MESSAGE = 66
    NOTIFY_CROSS_REF = 74
    NOTIFY_DOMAIN_KEY_UPDATE = 19
    NOTIFY_INSERTED = 73
    REGISTER = 64
    REQUEST_CLOSE_DOMAIN = 31
    REQUEST_COUNT_TRANSACTIONS = 95
    REQUEST_CROSS_REF_LIST = 92
    REQUEST_CROSS_REF_VERIFY = 90
    REQUEST_ECDH_KEY_EXCHANGE = 33
    REQUEST_GATHER_SIGNATURE = 67
    REQUEST_GET_CONFIG = 8
    REQUEST_GET_DOMAINLIST = 13
    REQUEST_GET_FORWARDING_LIST = 25
    REQUEST_GET_NEIGHBORLIST = 21
    REQUEST_GET_NODEID = 27
    REQUEST_GET_NOTIFICATION_LIST = 29
    REQUEST_GET_STATS = 17
    REQUEST_GET_USERS = 23
    REQUEST_INSERT = 71
    REQUEST_INSERT_NOTIFICATION = 15
    REQUEST_MANIP_LEDGER_SUBSYS = 10
    REQUEST_REGISTER_HASH_IN_SUBSYS = 128
    REQUEST_REPAIR = 94
    REQUEST_SEARCH_TRANSACTION = 82
    REQUEST_SEARCH_WITH_CONDITIONS = 86
    REQUEST_SETUP_DOMAIN = 0
```

```
REQUEST_SET_STATIC_NODE = 4
REQUEST_SIGNATURE = 69
REQUEST_TRAVERSE_TRANSACTIONS = 88
REQUEST_VERIFY_HASH_IN_SUBSYS = 130
RESPONSE_CLOSE_DOMAIN = 32
RESPONSE_COUNT_TRANSACTIONS = 95
RESPONSE_CROSS_REF_LIST = 93
RESPONSE_CROSS_REF_VERIFY = 91
RESPONSE_ECDH_KEY_EXCHANGE = 34
RESPONSE_GATHER_SIGNATURE = 68
RESPONSE_GET_CONFIG = 9
RESPONSE_GET_DOMAINLIST = 14
RESPONSE_GET_FORWARDING_LIST = 26
RESPONSE_GET_NEIGHBORLIST = 22
RESPONSE_GET_NODEID = 28
RESPONSE_GET_NOTIFICATION_LIST = 30
RESPONSE_GET_STATS = 18
RESPONSE_GET_USERS = 24
RESPONSE_INSERT = 72
RESPONSE_MANIP_LEDGER_SUBSYS = 11
RESPONSE_REGISTER_HASH_IN_SUBSYS = 129
RESPONSE_SEARCH_TRANSACTION = 83
RESPONSE_SEARCH_WITH_CONDITIONS = 87
RESPONSE_SETUP_DOMAIN = 1
RESPONSE_SET_STATIC_NODE = 5
RESPONSE_SIGNATURE = 70
RESPONSE_TRAVERSE_TRANSACTIONS = 89
RESPONSE_VERIFY_HASH_IN_SUBSYS = 131
UNREGISTER = 65
```

```
bbclib.compat.bbclib.add_event_asset(transaction, event_idx, asset_group_id, user_id, as-
                                   set_body=None, asset_file=None)
```

Utility to add BBcEvent object with BBcAsset in the transaction

```
bbclib.compat.bbclib.add_pointer_in_relation(relation, ref_transaction_id=None,
                                           ref_asset_id=None)
```

Utility to add BBcRelation object with BBcPointer in the BBcRelation object

```
bbclib.compat.bbclib.add_reference_to_transaction(transaction, asset_group_id,
                                                  ref_transaction_obj,
                                                  event_index_in_ref)
```

Utility to add BBcReference object in the transaction

**Returns****Return type** *BBcReference*

`bbclib.compat.bbclib.add_relation_asset` (*transaction*, *relation\_idx*, *asset\_group\_id*, *user\_id*,  
*asset\_body=None*, *asset\_file=None*)

Utility to add BBcRelation object with BBcAsset in the transaction

`bbclib.compat.bbclib.add_relation_pointer` (*transaction*, *relation\_idx*,  
*ref\_transaction\_id=None*, *ref\_asset\_id=None*)

Utility to add BBcRelation object with BBcPointer in the transaction

`bbclib.compat.bbclib.bin2str_base64` (*dat*)

`bbclib.compat.bbclib.convert_id_to_string` (*data*, *bytelen=32*)

Convert binary data to hex string

`bbclib.compat.bbclib.convert_idstring_to_bytes` (*datastr*, *bytelen=32*)

Convert hex string to binary data

`bbclib.compat.bbclib.deep_copy_with_key_stringify` (*u*, *d=None*)

Utility for updating nested dictionary

`bbclib.compat.bbclib.get_bigint` (*ptr*, *dat*)

`bbclib.compat.bbclib.get_n_byte_int` (*ptr*, *n*, *dat*)

`bbclib.compat.bbclib.get_n_bytes` (*ptr*, *n*, *dat*)

`bbclib.compat.bbclib.get_new_id` (*seed\_str=None*, *include\_timestamp=True*)

Return 256-bit binary data

**Parameters**

- **seed\_str** (*str*) – seed string that is hashed by SHA256
- **include\_timestamp** (*bool*) – if True, timestamp (current time) is appended to the seed string

**Returns** 256-bit binary**Return type** bytes

`bbclib.compat.bbclib.get_random_id` ()

Return 256-bit binary data

**Returns** 256-bit random binary**Return type** bytes

`bbclib.compat.bbclib.get_random_value` (*length=32*)

Return 1-byte random value

`bbclib.compat.bbclib.make_relation_with_asset` (*asset\_group\_id*, *user\_id*, *as-*  
*set\_body=None*, *asset\_file=None*,  
*format\_type=0*, *id\_length=32*)

Utility to make BBcRelation object

`bbclib.compat.bbclib.make_transaction` (*event\_num=0*, *relation\_num=0*, *witness=False*, *for-*  
*mat\_type=0*, *id\_length=32*)

Utility to make transaction object

**Parameters**

- **event\_num** (*int*) – the number of BBcEvent object to include in the transaction
- **relation\_num** (*int*) – the number of BBcRelation object to include in the transaction

- **witness** (*bool*) – If true, BBcWitness object is included in the transaction
- **format\_type** (*int*) – Data format defined in BBcFormat class
- **id\_length** (*int*) – If <32, IDs will be truncated

**Returns****Return type** *BBcTransaction*`bbclib.compat.bbclib.recover_signature_object (data, format_type=0)`

Deserialize signature data

`bbclib.compat.bbclib.reset_error ()``bbclib.compat.bbclib.set_error (code=-1, txt=“")``bbclib.compat.bbclib.str_binary (dat)``bbclib.compat.bbclib.to_1byte (val)``bbclib.compat.bbclib.to_2byte (val)``bbclib.compat.bbclib.to_4byte (val)``bbclib.compat.bbclib.to_8byte (val)``bbclib.compat.bbclib.to_bigint (val, size=32)``bbclib.compat.bbclib.validate_transaction_object (txobj, asset_files=None)`

Validate transaction and its asset

**Parameters**

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

**Returns** True if valid tuple: list of valid assets tuple: list of invalid assets**Return type** bool`bbclib.compat.bbclib.verify_using_cross_ref (domain_id, transaction_id, transaction_base_digest, cross_ref_data, sigdata, format_type=0)`

Confirm the existence of the transaction using cross\_ref

**Parameters**

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

**Returns** True if valid**Return type** bool



### 1.1.1.2 Module contents

## 1.1.2 bbclib.libs package

### 1.1.2.1 Submodules

#### bbclib.libs.bbclib\_asset module

Copyright (c) 2018 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 bbclib.libs.bbclib_asset.BBcAsset (user_id=None,      asset_file=None,      as-
                                         set_body=None, id_length=None, version=2)
```

Bases: object

Asset part in a transaction

```
add (user_id=None, asset_file=None, asset_body=None)
```

Add parts in this object

```
digest ()
```

Calculate the digest

The digest corresponds to the asset\_id of this object

**Returns** asset\_id (or digest)

**Return type** bytes

```
get_asset_file ()
```

Get asset file content and its digest

**Returns** digest of the file content bytes: the file content

**Return type** bytes

```
pack (for_digest_calculation=False)
```

Pack this object

**Parameters** **for\_digest\_calculation** (*bool*) – True if digest calculation

**Returns** packed binary data

**Return type** bytes

```
recover_asset_file (asset_file)
```

Recover asset file info from the given raw content

```
unpack (data)
```

Unpack into this object

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

**Returns** True if successful

**Return type** bool

## bbclib.libs.bbclib\_config module

Copyright (c) 2018 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.

## bbclib.libs.bbclib\_crossref module

Copyright (c) 2018 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 bbclib.libs.bbclib_crossref.BBcCrossRef (domain_id=None, transaction_id=None,  
                                              unpack=None)
```

Bases: object

CrossRef part in a transaction

```
pack ()
```

Pack this object

**Returns** packed binary data

**Return type** bytes

```
unpack (data)
```

Unpack into this object

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

**Returns** True if successful

**Return type** bool

## bbclib.libs.bbclib\_error module

Copyright (c) 2019 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.

## bbclib.libs.bbclib\_event module

Copyright (c) 2018 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** bbclib.libs.bbclib\_event.**BBcEvent** (*asset\_group\_id=None, id\_length=None*)

Bases: object

Event part in a transaction

**add** (*asset\_group\_id=None, reference\_index=None, mandatory\_approver=None, op-  
tion\_approver\_num\_numerator=0, option\_approver\_num\_denominator=0, op-  
tion\_approver=None, asset=None*)  
Add parts

**pack** ()

Pack this object

**Returns** packed binary data

**Return type** bytes

**unpack** (*data*)

Unpack into this object

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

**Returns** True if successful

**Return type** bool

## bbclib.libs.bbclib\_keypair module

Copyright (c) 2018 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** bbclib.libs.bbclib\_keypair.**KeyPair** (*curvetype=2, compression=False, privkey=None,  
pubkey=None*)

Bases: object

**POINT\_CONVERSION\_COMPRESSED** = 2

**POINT\_CONVERSION\_UNCOMPRESSED** = 4

Key pair container

**generate** ()

Generate a new key pair

**get\_private\_key\_in\_der()**  
Return private key in DER format

**get\_private\_key\_in\_pem()**  
Return private key in PEM format

**get\_public\_key\_in\_der()**  
Return private key in DER format

**get\_public\_key\_in\_pem()**  
Return public key in PEM format

**import\_publickey\_cert\_pem(cert\_pemstring, privkey\_pemstring=None)**  
Verify and import X509 public key certificate in pem format

**mk\_keyobj\_from\_private\_key()**  
Make a keypair object from the binary data of private key

**mk\_keyobj\_from\_private\_key\_der(derdat)**  
Make a keypair object from the private key in DER format

**mk\_keyobj\_from\_private\_key\_pem(pemdat\_string)**  
Make a keypair object from the private key in PEM format

**sign(digest)**  
Sign to the given value

**Parameters** **digest** (*bytes*) – given value

**Returns** signature

**Return type** bytes

**to\_binary(dat)**

**verify(digest, sig)**  
Verify the digest and the signature using the private key in this object

**class** bbclib.libs.bbclib\_keypair.**KeyType**  
Bases: object

**ECDSA\_P256v1** = 2

**ECDSA\_SECP256k1** = 1

**NOT\_INITIALIZED** = 0

## bbclib.libs.bbclib\_msgtype module

Copyright (c) 2018 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** bbclib.libs.bbclib\_msgtype.**MsgType**  
Bases: object

Message types for between core node and client

```
CANCEL_INSERT_NOTIFICATION = 16
DOMAIN_PING = 12
MESSAGE = 66
NOTIFY_CROSS_REF = 74
NOTIFY_DOMAIN_KEY_UPDATE = 19
NOTIFY_INSERTED = 73
REGISTER = 64
REQUEST_CLOSE_DOMAIN = 31
REQUEST_COUNT_TRANSACTIONS = 95
REQUEST_CROSS_REF_LIST = 92
REQUEST_CROSS_REF_VERIFY = 90
REQUEST_ECDH_KEY_EXCHANGE = 33
REQUEST_GATHER_SIGNATURE = 67
REQUEST_GET_CONFIG = 8
REQUEST_GET_DOMAINLIST = 13
REQUEST_GET_FORWARDING_LIST = 25
REQUEST_GET_NEIGHBORLIST = 21
REQUEST_GET_NODEID = 27
REQUEST_GET_NOTIFICATION_LIST = 29
REQUEST_GET_STATS = 17
REQUEST_GET_USERS = 23
REQUEST_INSERT = 71
REQUEST_INSERT_NOTIFICATION = 15
REQUEST_MANIP_LEDGER_SUBSYS = 10
REQUEST_REGISTER_HASH_IN_SUBSYS = 128
REQUEST_REPAIR = 94
REQUEST_SEARCH_TRANSACTION = 82
REQUEST_SEARCH_WITH_CONDITIONS = 86
REQUEST_SETUP_DOMAIN = 0
REQUEST_SET_STATIC_NODE = 4
REQUEST_SIGNATURE = 69
REQUEST_TRAVERSE_TRANSACTIONS = 88
REQUEST_VERIFY_HASH_IN_SUBSYS = 130
RESPONSE_CLOSE_DOMAIN = 32
RESPONSE_COUNT_TRANSACTIONS = 95
RESPONSE_CROSS_REF_LIST = 93
```

```
RESPONSE_CROSS_REF_VERIFY = 91
RESPONSE_ECDH_KEY_EXCHANGE = 34
RESPONSE_GATHER_SIGNATURE = 68
RESPONSE_GET_CONFIG = 9
RESPONSE_GET_DOMAINLIST = 14
RESPONSE_GET_FORWARDING_LIST = 26
RESPONSE_GET_NEIGHBORLIST = 22
RESPONSE_GET_NODEID = 28
RESPONSE_GET_NOTIFICATION_LIST = 30
RESPONSE_GET_STATS = 18
RESPONSE_GET_USERS = 24
RESPONSE_INSERT = 72
RESPONSE_MANIP_LEDGER_SUBSYS = 11
RESPONSE_REGISTER_HASH_IN_SUBSYS = 129
RESPONSE_SEARCH_TRANSACTION = 83
RESPONSE_SEARCH_WITH_CONDITIONS = 87
RESPONSE_SETUP_DOMAIN = 1
RESPONSE_SET_STATIC_NODE = 5
RESPONSE_SIGNATURE = 70
RESPONSE_TRAVERSE_TRANSACTIONS = 89
RESPONSE_VERIFY_HASH_IN_SUBSYS = 131
UNREGISTER = 65
```

### bbclib.libs.bbclib\_pointer module

Copyright (c) 2018 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 bbclib.libs.bbclib_pointer.BBcPointer(transaction_id=None, asset_id=None,  
                                           id_length=None)
```

Bases: object

Pointer part in a transaction

```
add(transaction_id=None, asset_id=None)
```

Add parts

**pack** ()  
 Pack this object

**Returns** packed binary data

**Return type** bytes

**unpack** (*data*)  
 Unpack into this object

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

**Returns** True if successful

**Return type** bool

## bbclib.libs.bbclib\_reference module

Copyright (c) 2018 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 bbclib.libs.bbclib_reference.BBcReference (asset_group_id, transaction, ref_transaction=None,
                                                event_index_in_ref=0,
                                                id_length=None)
```

Bases: object

Reference part in a transaction

**add\_signature** (*user\_id=None*, *signature=None*)  
 Add signature in the reserved space

**Parameters**

- **user\_id** (*bytes*) – user\_id of the signature owner
- **signature** (*BBcSignature*) – signature

**get\_destinations** ()  
 Return the list of approvers in the referred transaction

**pack** ()  
 Pack this object

**Returns** packed binary data

**Return type** bytes

**prepare\_reference** (*ref\_transaction*)  
 Read the previous referencing transaction

**unpack** (*data*)  
 unpack into this object

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

**Returns** True if successful

**Return type** bool

### bbclib.libs.bbclib\_relation module

Copyright (c) 2018 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** `bbclib.libs.bbclib_relation.BBcRelation` (*asset\_group\_id=None, id\_length=None*)

Bases: `object`

Relation part in a transaction

**add** (*asset\_group\_id=None, asset=None, pointer=None*)

Add parts

**pack** ()

Pack this object

**Returns** packed binary data

**Return type** bytes

**unpack** (*data*)

Unpack data into transaction object

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

**Returns** True if successful

**Return type** bool

### bbclib.libs.bbclib\_signature module

Copyright (c) 2018 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** `bbclib.libs.bbclib_signature.BBcSignature` (*key\_type=2, unpack=None*)

Bases: `object`

Signature part in a transaction

**add** (*signature=None, pubkey=None*)

Add signature and public key

**pack** ()

Pack this object



**unpack** (*data*)

Unpack into this object

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

**Returns** True if successful

**Return type** bool

**verify** (*digest*, *pubkey=None*)

Verify digest using pubkey in signature

**Parameters**

- **digest** (*bytes*) – digest to verify
- **pubkey** (*bytes*) – external public key for verification

**Returns** 0:invalid, 1:valid

**Return type** int

## bbclib.libs.bbclib\_transaction 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** bbclib.libs.bbclib\_transaction.**BBcTransaction** (*version=1*, *unpack=None*,  
*id\_length=None*)

Bases: object

Transaction object

**WITH\_WIRE** = **False**

**add** (*event=None*, *reference=None*, *relation=None*, *witness=None*, *cross\_ref=None*)

Add parts

**add\_signature** (*user\_id=None*, *signature=None*)

Add signature in the reserved space

**Parameters**

- **user\_id** (*bytes*) – user\_id of the signature owner
- **signature** (*BBcSignature*) – signature

**Returns** True if successful

**Return type** bool

**digest** ()

Calculate the digest

The digest corresponds to the transaction\_id of this object

**Returns** transaction\_id (or digest)

**Return type** bytes

**get\_sig\_index** (*user\_id*)

Reserve a space for signature for the specified user\_id

**Parameters** **user\_id** (*bytes*) – user\_id whose signature will be added to the signature part

**Returns** position (index) in the signature part

**Return type** int

**pack** (*for\_id=False*)

Pack the whole parts

**set\_sig\_index** (*user\_id, idx*)

Map a user\_id with the index of signature list

**Parameters**

- **user\_id** (*bytes*) – user\_id whose signature will be added to the signature part
- **idx** (*int*) – index number

**sign** (*key\_type=2, private\_key=None, public\_key=None, keypair=None, no\_pubkey=False*)

Sign the transaction

**Parameters**

- **key\_type** (*int*) – Type of encryption key's curve
- **private\_key** (*bytes*) –
- **public\_key** (*bytes*) –
- **keypair** (*KeyPair*) – keypair or set of private\_key and public\_key needs to be given
- **no\_pubkey** (*bool*) – If True, public key is not contained in the BBcSignature object (needs to be given externally when verification)

**Returns**

**Return type** *BBcSignature*

**unpack** (*data*)

Unpack into this object

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

**Returns** True if successful

**Return type** bool

## bbclib.libs.bbclib\_utils module

Copyright (c) 2018 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.

`bbclib.libs.bbclib_utils.add_event_asset(transaction, event_idx, asset_group_id, user_id, asset_body=None, asset_file=None)`

Utility to add BBcEvent object with BBcAsset in the transaction

`bbclib.libs.bbclib_utils.add_pointer_in_relation(relation, ref_transaction_id=None, ref_asset_id=None)`

Utility to add BBcRelation object with BBcPointer in the BBcRelation object

`bbclib.libs.bbclib_utils.add_reference_to_transaction(transaction, asset_group_id, ref_transaction_obj, event_index_in_ref)`

Utility to add BBcReference object in the transaction

#### Returns

**Return type** *BBcReference*

`bbclib.libs.bbclib_utils.add_relation_asset(transaction, relation_idx, asset_group_id, user_id, asset_body=None, asset_file=None)`

Utility to add BBcRelation object with BBcAsset in the transaction

`bbclib.libs.bbclib_utils.add_relation_pointer(transaction, relation_idx, ref_transaction_id=None, ref_asset_id=None)`

Utility to add BBcRelation object with BBcPointer in the transaction

`bbclib.libs.bbclib_utils.bin2str_base64(dat)`

`bbclib.libs.bbclib_utils.convert_id_to_string(data, bytelen=32)`

Convert binary data to hex string

`bbclib.libs.bbclib_utils.convert_idstring_to_bytes(datastr, bytelen=32)`

Convert hex string to binary data

`bbclib.libs.bbclib_utils.deep_copy_with_key_stringify(u, d=None)`

Utility for updating nested dictionary

`bbclib.libs.bbclib_utils.get_bigint(ptr, dat)`

`bbclib.libs.bbclib_utils.get_n_byte_int(ptr, n, dat)`

`bbclib.libs.bbclib_utils.get_n_bytes(ptr, n, dat)`

`bbclib.libs.bbclib_utils.get_new_id(seed_str=None, include_timestamp=True)`

Return 256-bit binary data

#### Parameters

- **seed\_str** (*str*) – seed string that is hashed by SHA256
- **include\_timestamp** (*bool*) – if True, timestamp (current time) is appended to the seed string

**Returns** 256-bit binary

**Return type** bytes

`bbclib.libs.bbclib_utils.get_random_id()`

Return 256-bit binary data

**Returns** 256-bit random binary

**Return type** bytes

`bbclib.libs.bbclib_utils.get_random_value (length=32)`

Return 1-byte random value

`bbclib.libs.bbclib_utils.make_relation_with_asset (asset_group_id, user_id, asset_body=None, asset_file=None)`

Utility to make BBcRelation object

`bbclib.libs.bbclib_utils.make_transaction (event_num=0, relation_num=0, witness=False, id_length=32)`

Utility to make transaction object

#### Parameters

- **event\_num** (*int*) – the number of BBcEvent object to include in the transaction
- **relation\_num** (*int*) – the number of BBcRelation object to include in the transaction
- **witness** (*bool*) – If true, BBcWitness object is included in the transaction
- **id\_length** (*int*) – If <32, IDs will be truncated (this params is for the backward compatibility)

#### Returns

Return type *BBcTransaction*

`bbclib.libs.bbclib_utils.recover_signature_object (data)`

Unpack signature data

`bbclib.libs.bbclib_utils.str_binary (dat)`

`bbclib.libs.bbclib_utils.to_1byte (val)`

`bbclib.libs.bbclib_utils.to_2byte (val)`

`bbclib.libs.bbclib_utils.to_4byte (val)`

`bbclib.libs.bbclib_utils.to_8byte (val)`

`bbclib.libs.bbclib_utils.to_bigint (val, size=32)`

`bbclib.libs.bbclib_utils.validate_transaction_object (txobj, asset_files=None)`

Validate transaction and its asset

#### Parameters

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

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

Return type `bool`

`bbclib.libs.bbclib_utils.verify_using_cross_ref (domain_id, transaction_id, transaction_base_digest, cross_ref_data, sig_data)`

Confirm the existence of the transaction using cross\_ref

#### Parameters

- **domain\_id** (*bytes*) – target domain\_id
- **transaction\_id** (*bytes*) – target transaction\_id of which existence you want to confirm
- **transaction\_base\_digest** (*bytes*) – digest obtained from the outer domain

- **cross\_ref\_data** (*bytes*) – packed BBcCrossRef object
- **sigdata** (*bytes*) – packed signature

**Returns** True if valid

**Return type** bool

### bbclib.libs.bbclib\_wire module

Copyright (c) 2018 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** bbclib.libs.bbclib\_wire.BBcFormat

Bases: object

**FORMAT\_PLAIN** = 0

**FORMAT\_ZLIB** = 16

**classmethod** generate (*txobj*, *format\_type=0*)

Transform transaction object in wire format

**Parameters**

- **txobj** – BBcTransaction object
- **format\_type** – 2-byte value of BBcFormat type

**Returns** binary data

**classmethod** strip (*data*)

Strip 2-byte wire header and recover plain binary

**Parameters** **data** – binary data with wire header

**Returns** plain binary data without the header

### bbclib.libs.bbclib\_witness module

Copyright (c) 2018 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** bbclib.libs.bbclib\_witness.BBcWitness (*id\_length=None*)

Bases: object

Witness part in a transaction

**add\_signature** (*user\_id=None, signature=None*)

Add signature in the reserved space for the *user\_id* that was registered before

**Parameters**

- **user\_id** (*bytes*) – *user\_id* of the signature owner
- **signature** (*BBcSignature*) – signature

**add\_witness** (*user\_id*)

Register *user\_id* in the list

**pack** ()

Pack this object

**Returns** packed binary data

**Return type** *bytes*

**unpack** (*data*)

Unpack into this object

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

**Returns** True if successful

**Return type** *bool*

### 1.1.2.2 Module contents

## 1.2 Module contents

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.

**bbclib.configure\_id\_length** (*conf*)

**bbclib.configure\_id\_length\_all** (*value*)

**bbclib.deserialize** (*txdata*)

Deserialize binary data with 2-byte wire header

**Parameters** **txdata** –

**Returns** *BBcTransaction*: *BBcTransaction* object int: 2-byte value of *BBcFormat* type

**bbclib.serialize** (*txobj, format\_type=0*)

Serialize transaction object with 2-byte wire header

**Parameters**

- **txobj** – *BBcTransaction* object
- **format\_type** – value defined in *bbclib\_wire.BBcFormat*

**Returns** *binary*

## CHAPTER 2

---

### Indices and tables

---

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





### b

- `bbclib`, 26
- `bbclib.compat`, 13
- `bbclib.compat.bbclib`, 1
- `bbclib.libs`, 26
  - `bbclib.asset`, 13
  - `bbclib.config`, 14
  - `bbclib.crossref`, 14
  - `bbclib.error`, 14
  - `bbclib.event`, 15
  - `bbclib.keypair`, 15
  - `bbclib.msgtype`, 16
  - `bbclib.pointer`, 18
  - `bbclib.reference`, 19
  - `bbclib.relation`, 20
  - `bbclib.signature`, 20
  - `bbclib.transaction`, 21
  - `bbclib.utils`, 22
  - `bbclib.wire`, 25
  - `bbclib.witness`, 25



## A

`add()` (*bbclib.compat.bbclib.BBcAsset* method), 1  
`add()` (*bbclib.compat.bbclib.BBcEvent* method), 3  
`add()` (*bbclib.compat.bbclib.BBcPointer* method), 3  
`add()` (*bbclib.compat.bbclib.BBcRelation* method), 5  
`add()` (*bbclib.compat.bbclib.BBcSignature* method), 5  
`add()` (*bbclib.compat.bbclib.BBcTransaction* method), 6  
`add()` (*bbclib.libs.bbclib\_asset.BBcAsset* method), 13  
`add()` (*bbclib.libs.bbclib\_event.BBcEvent* method), 15  
`add()` (*bbclib.libs.bbclib\_pointer.BBcPointer* method), 18  
`add()` (*bbclib.libs.bbclib\_relation.BBcRelation* method), 20  
`add()` (*bbclib.libs.bbclib\_signature.BBcSignature* method), 20  
`add()` (*bbclib.libs.bbclib\_transaction.BBcTransaction* method), 21  
`add_event_asset()` (in module *bbclib.compat.bbclib*), 10  
`add_event_asset()` (in module *bbclib.libs.bbclib\_utils*), 22  
`add_pointer_in_relation()` (in module *bbclib.compat.bbclib*), 10  
`add_pointer_in_relation()` (in module *bbclib.libs.bbclib\_utils*), 23  
`add_reference_to_transaction()` (in module *bbclib.compat.bbclib*), 10  
`add_reference_to_transaction()` (in module *bbclib.libs.bbclib\_utils*), 23  
`add_relation_asset()` (in module *bbclib.compat.bbclib*), 11  
`add_relation_asset()` (in module *bbclib.libs.bbclib\_utils*), 23  
`add_relation_pointer()` (in module *bbclib.compat.bbclib*), 11  
`add_relation_pointer()` (in module *bbclib.libs.bbclib\_utils*), 23  
`add_signature()`

*bbclib.compat.bbclib.BBcReference* method), 4  
`add_signature()` (*bbclib.compat.bbclib.BBcTransaction* method), 6  
`add_signature()` (*bbclib.compat.bbclib.BBcWitness* method), 7  
`add_signature()` (*bbclib.libs.bbclib\_reference.BBcReference* method), 19  
`add_signature()` (*bbclib.libs.bbclib\_transaction.BBcTransaction* method), 21  
`add_signature()` (*bbclib.libs.bbclib\_witness.BBcWitness* method), 25  
`add_witness()` (*bbclib.compat.bbclib.BBcWitness* method), 7  
`add_witness()` (*bbclib.libs.bbclib\_witness.BBcWitness* method), 26

## B

*BBcAsset* (class in *bbclib.compat.bbclib*), 1  
*BBcAsset* (class in *bbclib.libs.bbclib\_asset*), 13  
*BBcCrossRef* (class in *bbclib.compat.bbclib*), 2  
*BBcCrossRef* (class in *bbclib.libs.bbclib\_crossref*), 14  
*BBcEvent* (class in *bbclib.compat.bbclib*), 3  
*BBcEvent* (class in *bbclib.libs.bbclib\_event*), 15  
*BBcFormat* (class in *bbclib.compat.bbclib*), 3  
*BBcFormat* (class in *bbclib.libs.bbclib\_wire*), 25  
*bbclib* (module), 26  
*bbclib.compat* (module), 13  
*bbclib.compat.bbclib* (module), 1  
*bbclib.libs* (module), 26  
*bbclib.libs.bbclib\_asset* (module), 13  
*bbclib.libs.bbclib\_config* (module), 14  
*bbclib.libs.bbclib\_crossref* (module), 14  
*bbclib.libs.bbclib\_error* (module), 14  
*bbclib.libs.bbclib\_event* (module), 15

bbclib.libs.bbclib\_keypair (module), 15  
 bbclib.libs.bbclib\_msgtype (module), 16  
 bbclib.libs.bbclib\_pointer (module), 18  
 bbclib.libs.bbclib\_reference (module), 19  
 bbclib.libs.bbclib\_relation (module), 20  
 bbclib.libs.bbclib\_signature (module), 20  
 bbclib.libs.bbclib\_transaction (module), 21  
 bbclib.libs.bbclib\_utils (module), 22  
 bbclib.libs.bbclib\_wire (module), 25  
 bbclib.libs.bbclib\_witness (module), 25  
 BBcPointer (class in bbclib.compat.bbclib), 3  
 BBcPointer (class in bbclib.libs.bbclib\_pointer), 18  
 BBcReference (class in bbclib.compat.bbclib), 4  
 BBcReference (class in bbclib.libs.bbclib\_reference), 19  
 BBcRelation (class in bbclib.compat.bbclib), 5  
 BBcRelation (class in bbclib.libs.bbclib\_relation), 20  
 BBcSignature (class in bbclib.compat.bbclib), 5  
 BBcSignature (class in bbclib.libs.bbclib\_signature), 20  
 BBcTransaction (class in bbclib.compat.bbclib), 6  
 BBcTransaction (class in bbclib.libs.bbclib\_transaction), 21  
 BBcWitness (class in bbclib.compat.bbclib), 7  
 BBcWitness (class in bbclib.libs.bbclib\_witness), 25  
 bin2str\_base64() (in module bbclib.compat.bbclib), 11  
 bin2str\_base64() (in module bbclib.libs.bbclib\_utils), 23

## C

CANCEL\_INSERT\_NOTIFICATION (bbclib.compat.bbclib.MsgType attribute), 9  
 CANCEL\_INSERT\_NOTIFICATION (bbclib.libs.bbclib\_msgtype.MsgType attribute), 16  
 configure\_id\_length() (in module bbclib), 26  
 configure\_id\_length\_all() (in module bbclib), 26  
 convert\_id\_to\_string() (in module bbclib.compat.bbclib), 11  
 convert\_id\_to\_string() (in module bbclib.libs.bbclib\_utils), 23  
 convert\_idstring\_to\_bytes() (in module bbclib.compat.bbclib), 11  
 convert\_idstring\_to\_bytes() (in module bbclib.libs.bbclib\_utils), 23

## D

deep\_copy\_with\_key\_stringify() (in module bbclib.compat.bbclib), 11  
 deep\_copy\_with\_key\_stringify() (in module bbclib.libs.bbclib\_utils), 23

deserialize() (bbclib.compat.bbclib.BBcAsset method), 1  
 deserialize() (bbclib.compat.bbclib.BBcCrossRef method), 2  
 deserialize() (bbclib.compat.bbclib.BBcEvent method), 3  
 deserialize() (bbclib.compat.bbclib.BBcPointer method), 4  
 deserialize() (bbclib.compat.bbclib.BBcReference method), 4  
 deserialize() (bbclib.compat.bbclib.BBcRelation method), 5  
 deserialize() (bbclib.compat.bbclib.BBcSignature method), 5  
 deserialize() (bbclib.compat.bbclib.BBcTransaction method), 6  
 deserialize() (bbclib.compat.bbclib.BBcWitness method), 7  
 deserialize() (in module bbclib), 26  
 deserialize\_obj() (bbclib.compat.bbclib.BBcAsset method), 2  
 deserialize\_obj() (bbclib.compat.bbclib.BBcCrossRef method), 2  
 deserialize\_obj() (bbclib.compat.bbclib.BBcEvent method), 3  
 deserialize\_obj() (bbclib.compat.bbclib.BBcPointer method), 4  
 deserialize\_obj() (bbclib.compat.bbclib.BBcReference method), 4  
 deserialize\_obj() (bbclib.compat.bbclib.BBcRelation method), 5  
 deserialize\_obj() (bbclib.compat.bbclib.BBcSignature method), 6  
 deserialize\_obj() (bbclib.compat.bbclib.BBcTransaction method), 6  
 deserialize\_obj() (bbclib.compat.bbclib.BBcWitness method), 7  
 digest() (bbclib.compat.bbclib.BBcAsset method), 2  
 digest() (bbclib.compat.bbclib.BBcTransaction method), 6  
 digest() (bbclib.libs.bbclib\_asset.BBcAsset method), 13  
 digest() (bbclib.libs.bbclib\_transaction.BBcTransaction method), 21  
 DOMAIN\_PING (bbclib.compat.bbclib.MsgType attribute), 9

DOMAIN\_PING (*bbclib.libs.bbclib\_msgtype.MsgType* attribute), 17

## E

ECDSA\_P256v1 (*bbclib.compat.bbclib.KeyType* attribute), 9

ECDSA\_P256v1 (*bbclib.libs.bbclib\_keypair.KeyType* attribute), 16

ECDSA\_SECP256k1 (*bbclib.compat.bbclib.KeyType* attribute), 9

ECDSA\_SECP256k1 (*bbclib.libs.bbclib\_keypair.KeyType* attribute), 16

## F

FORMAT\_BINARY (*bbclib.compat.bbclib.BBcFormat* attribute), 3

FORMAT\_BSON (*bbclib.compat.bbclib.BBcFormat* attribute), 3

FORMAT\_BSON\_COMPRESS\_BZ2 (*bbclib.compat.bbclib.BBcFormat* attribute), 3

FORMAT\_BSON\_COMPRESS\_ZLIB (*bbclib.compat.bbclib.BBcFormat* attribute), 3

FORMAT\_MSGPACK (*bbclib.compat.bbclib.BBcFormat* attribute), 3

FORMAT\_MSGPACK\_COMPRESS\_BZ2 (*bbclib.compat.bbclib.BBcFormat* attribute), 3

FORMAT\_MSGPACK\_COMPRESS\_ZLIB (*bbclib.compat.bbclib.BBcFormat* attribute), 3

FORMAT\_PLAIN (*bbclib.libs.bbclib\_wire.BBcFormat* attribute), 25

FORMAT\_ZLIB (*bbclib.libs.bbclib\_wire.BBcFormat* attribute), 25

## G

generate() (*bbclib.compat.bbclib.KeyPair* method), 8

generate() (*bbclib.libs.bbclib\_keypair.KeyPair* method), 15

generate() (*bbclib.libs.bbclib\_wire.BBcFormat* class method), 25

get\_asset\_file() (*bbclib.compat.bbclib.BBcAsset* method), 2

get\_asset\_file() (*bbclib.libs.bbclib\_asset.BBcAsset* method), 13

get\_bigint() (*in module bbclib.compat.bbclib*), 11

get\_bigint() (*in module bbclib.libs.bbclib\_utils*), 23

get\_destinations() (*bbclib.compat.bbclib.BBcReference* method), 4

get\_destinations() (*bbclib.libs.bbclib\_reference.BBcReference* method), 19

get\_dict() (*bbclib.compat.bbclib.BBcAsset* method), 2

get\_dict() (*bbclib.compat.bbclib.BBcCrossRef* method), 2

get\_dict() (*bbclib.compat.bbclib.BBcEvent* method), 3

get\_dict() (*bbclib.compat.bbclib.BBcPointer* method), 4

get\_dict() (*bbclib.compat.bbclib.BBcReference* method), 4

get\_dict() (*bbclib.compat.bbclib.BBcRelation* method), 5

get\_dict() (*bbclib.compat.bbclib.BBcSignature* method), 6

get\_dict() (*bbclib.compat.bbclib.BBcWitness* method), 8

get\_n\_byte\_int() (*in module bbclib.compat.bbclib*), 11

get\_n\_byte\_int() (*in module bbclib.libs.bbclib\_utils*), 23

get\_n\_bytes() (*in module bbclib.compat.bbclib*), 11

get\_n\_bytes() (*in module bbclib.libs.bbclib\_utils*), 23

get\_new\_id() (*in module bbclib.compat.bbclib*), 11

get\_new\_id() (*in module bbclib.libs.bbclib\_utils*), 23

get\_private\_key\_in\_der() (*bbclib.compat.bbclib.KeyPair* method), 8

get\_private\_key\_in\_der() (*bbclib.libs.bbclib\_keypair.KeyPair* method), 15

get\_private\_key\_in\_pem() (*bbclib.compat.bbclib.KeyPair* method), 8

get\_private\_key\_in\_pem() (*bbclib.libs.bbclib\_keypair.KeyPair* method), 16

get\_public\_key\_in\_der() (*bbclib.libs.bbclib\_keypair.KeyPair* method), 16

get\_public\_key\_in\_pem() (*bbclib.compat.bbclib.KeyPair* method), 8

get\_public\_key\_in\_pem() (*bbclib.libs.bbclib\_keypair.KeyPair* method), 16

get\_random\_id() (*in module bbclib.compat.bbclib*), 11

get\_random\_id() (*in module bbclib.libs.bbclib\_utils*), 23

get\_random\_value() (*in module bbclib.compat.bbclib*), 11

get\_random\_value() (*in module bbclib.libs.bbclib\_utils*), 23

`get_referred_transaction()` (*bbclib.compat.bbclib.BBcReference* method), 5

`get_sig_index()` (*bbclib.compat.bbclib.BBcTransaction* method), 7

`get_sig_index()` (*bbclib.libs.bbclib\_transaction.BBcTransaction* method), 22

## I

`import_publickey_cert_pem()` (*bbclib.compat.bbclib.KeyPair* method), 8

`import_publickey_cert_pem()` (*bbclib.libs.bbclib\_keypair.KeyPair* method), 16

## K

`KeyPair` (class in *bbclib.compat.bbclib*), 8

`KeyPair` (class in *bbclib.libs.bbclib\_keypair*), 15

`KeyType` (class in *bbclib.compat.bbclib*), 8

`KeyType` (class in *bbclib.libs.bbclib\_keypair*), 16

## M

`make_relation_with_asset()` (in module *bbclib.compat.bbclib*), 11

`make_relation_with_asset()` (in module *bbclib.libs.bbclib\_utils*), 24

`make_transaction()` (in module *bbclib.compat.bbclib*), 11

`make_transaction()` (in module *bbclib.libs.bbclib\_utils*), 24

`MESSAGE` (*bbclib.compat.bbclib.MsgType* attribute), 9

`MESSAGE` (*bbclib.libs.bbclib\_msgtype.MsgType* attribute), 17

`mk_keyobj_from_private_key()` (*bbclib.compat.bbclib.KeyPair* method), 8

`mk_keyobj_from_private_key()` (*bbclib.libs.bbclib\_keypair.KeyPair* method), 16

`mk_keyobj_from_private_key_der()` (*bbclib.compat.bbclib.KeyPair* method), 8

`mk_keyobj_from_private_key_der()` (*bbclib.libs.bbclib\_keypair.KeyPair* method), 16

`mk_keyobj_from_private_key_pem()` (*bbclib.compat.bbclib.KeyPair* method), 8

`mk_keyobj_from_private_key_pem()` (*bbclib.libs.bbclib\_keypair.KeyPair* method), 16

`MsgType` (class in *bbclib.compat.bbclib*), 9

`MsgType` (class in *bbclib.libs.bbclib\_msgtype*), 16

## N

`NOT_INITIALIZED` (*bbclib.compat.bbclib.KeyType* attribute), 9

`NOT_INITIALIZED` (*bbclib.libs.bbclib\_keypair.KeyType* attribute), 16

`NOTIFY_CROSS_REF` (*bbclib.compat.bbclib.MsgType* attribute), 9

`NOTIFY_CROSS_REF` (*bbclib.libs.bbclib\_msgtype.MsgType* attribute), 17

`NOTIFY_DOMAIN_KEY_UPDATE` (*bbclib.compat.bbclib.MsgType* attribute), 9

`NOTIFY_DOMAIN_KEY_UPDATE` (*bbclib.libs.bbclib\_msgtype.MsgType* attribute), 17

`NOTIFY_INSERTED` (*bbclib.compat.bbclib.MsgType* attribute), 9

`NOTIFY_INSERTED` (*bbclib.libs.bbclib\_msgtype.MsgType* attribute), 17

## P

`pack()` (*bbclib.libs.bbclib\_asset.BBcAsset* method), 13

`pack()` (*bbclib.libs.bbclib\_crossref.BBcCrossRef* method), 14

`pack()` (*bbclib.libs.bbclib\_event.BBcEvent* method), 15

`pack()` (*bbclib.libs.bbclib\_pointer.BBcPointer* method), 18

`pack()` (*bbclib.libs.bbclib\_reference.BBcReference* method), 19

`pack()` (*bbclib.libs.bbclib\_relation.BBcRelation* method), 20

`pack()` (*bbclib.libs.bbclib\_signature.BBcSignature* method), 20

`pack()` (*bbclib.libs.bbclib\_transaction.BBcTransaction* method), 22

`pack()` (*bbclib.libs.bbclib\_witness.BBcWitness* method), 26

`POINT_CONVERSION_COMPRESSED` (*bbclib.compat.bbclib.KeyPair* attribute), 8

`POINT_CONVERSION_COMPRESSED` (*bbclib.libs.bbclib\_keypair.KeyPair* attribute), 15

`POINT_CONVERSION_UNCOMPRESSED` (*bbclib.compat.bbclib.KeyPair* attribute), 8

`POINT_CONVERSION_UNCOMPRESSED` (*bbclib.libs.bbclib\_keypair.KeyPair* attribute), 15

`prepare_reference()` (*bbclib.compat.bbclib.BBcReference* method), 5

`prepare_reference()` (*bbclib.libs.bbclib\_reference.BBcReference* method), 16

method), 19

## R

recover\_asset\_file() (bb-clib.compat.bbclib.BBcAsset method), 2

recover\_asset\_file() (bb-clib.libs.bbclib\_asset.BBcAsset method), 13

recover\_signature\_object() (in module bb-clib.compat.bbclib), 12

recover\_signature\_object() (in module bb-clib.libs.bbclib\_utils), 24

REGISTER (bbclib.compat.bbclib.MsgType attribute), 9

REGISTER (bbclib.libs.bbclib\_msgtype.MsgType attribute), 17

REQUEST\_CLOSE\_DOMAIN (bb-clib.compat.bbclib.MsgType attribute), 9

REQUEST\_CLOSE\_DOMAIN (bb-clib.libs.bbclib\_msgtype.MsgType attribute), 17

REQUEST\_COUNT\_TRANSACTIONS (bb-clib.compat.bbclib.MsgType attribute), 9

REQUEST\_COUNT\_TRANSACTIONS (bb-clib.libs.bbclib\_msgtype.MsgType attribute), 17

REQUEST\_CROSS\_REF\_LIST (bb-clib.compat.bbclib.MsgType attribute), 9

REQUEST\_CROSS\_REF\_LIST (bb-clib.libs.bbclib\_msgtype.MsgType attribute), 17

REQUEST\_CROSS\_REF\_VERIFY (bb-clib.compat.bbclib.MsgType attribute), 9

REQUEST\_CROSS\_REF\_VERIFY (bb-clib.libs.bbclib\_msgtype.MsgType attribute), 17

REQUEST\_ECDH\_KEY\_EXCHANGE (bb-clib.compat.bbclib.MsgType attribute), 9

REQUEST\_ECDH\_KEY\_EXCHANGE (bb-clib.libs.bbclib\_msgtype.MsgType attribute), 17

REQUEST\_GATHER\_SIGNATURE (bb-clib.compat.bbclib.MsgType attribute), 9

REQUEST\_GATHER\_SIGNATURE (bb-clib.libs.bbclib\_msgtype.MsgType attribute), 17

REQUEST\_GET\_CONFIG (bb-clib.compat.bbclib.MsgType attribute), 9

REQUEST\_GET\_CONFIG (bb-clib.libs.bbclib\_msgtype.MsgType attribute), 17

REQUEST\_GET\_DOMAINLIST (bb-clib.compat.bbclib.MsgType attribute), 9

REQUEST\_GET\_DOMAINLIST (bb-clib.libs.bbclib\_msgtype.MsgType attribute),

17

REQUEST\_GET\_FORWARDING\_LIST (bb-clib.compat.bbclib.MsgType attribute), 9

REQUEST\_GET\_FORWARDING\_LIST (bb-clib.libs.bbclib\_msgtype.MsgType attribute), 17

REQUEST\_GET\_NEIGHBORLIST (bb-clib.compat.bbclib.MsgType attribute), 9

REQUEST\_GET\_NEIGHBORLIST (bb-clib.libs.bbclib\_msgtype.MsgType attribute), 17

REQUEST\_GET\_NODEID (bb-clib.compat.bbclib.MsgType attribute), 9

REQUEST\_GET\_NODEID (bb-clib.libs.bbclib\_msgtype.MsgType attribute), 17

REQUEST\_GET\_NOTIFICATION\_LIST (bb-clib.compat.bbclib.MsgType attribute), 9

REQUEST\_GET\_NOTIFICATION\_LIST (bb-clib.libs.bbclib\_msgtype.MsgType attribute), 17

REQUEST\_GET\_STATS (bbclib.compat.bbclib.MsgType attribute), 9

REQUEST\_GET\_STATS (bb-clib.libs.bbclib\_msgtype.MsgType attribute), 17

REQUEST\_GET\_USERS (bbclib.compat.bbclib.MsgType attribute), 9

REQUEST\_GET\_USERS (bb-clib.libs.bbclib\_msgtype.MsgType attribute), 17

REQUEST\_INSERT (bbclib.compat.bbclib.MsgType attribute), 9

REQUEST\_INSERT (bb-clib.libs.bbclib\_msgtype.MsgType attribute), 17

REQUEST\_INSERT\_NOTIFICATION (bb-clib.compat.bbclib.MsgType attribute), 9

REQUEST\_INSERT\_NOTIFICATION (bb-clib.libs.bbclib\_msgtype.MsgType attribute), 17

REQUEST\_MANIP\_LEDGER\_SUBSYS (bb-clib.compat.bbclib.MsgType attribute), 9

REQUEST\_MANIP\_LEDGER\_SUBSYS (bb-clib.libs.bbclib\_msgtype.MsgType attribute), 17

REQUEST\_REGISTER\_HASH\_IN\_SUBSYS (bb-clib.compat.bbclib.MsgType attribute), 9

REQUEST\_REGISTER\_HASH\_IN\_SUBSYS (bb-clib.libs.bbclib\_msgtype.MsgType attribute), 17

REQUEST\_REPAIR (bbclib.compat.bbclib.MsgType attribute), 9

REQUEST\_REPAIR (bb-clib.libs.bbclib\_msgtype.MsgType attribute),



<i>clib.libs.bbclib_msgtype.MsgType</i> attribute), 17	<i>clib.compat.bbclib.MsgType</i> attribute), 10
REQUEST_SEARCH_TRANSACTION ( <i>bb-clib.compat.bbclib.MsgType</i> attribute), 9	RESPONSE_CROSS_REF_VERIFY ( <i>bb-clib.libs.bbclib_msgtype.MsgType</i> attribute), 17
REQUEST_SEARCH_TRANSACTION ( <i>bb-clib.libs.bbclib_msgtype.MsgType</i> attribute), 17	RESPONSE_ECDH_KEY_EXCHANGE ( <i>bb-clib.compat.bbclib.MsgType</i> attribute), 10
REQUEST_SEARCH_WITH_CONDITIONS ( <i>bb-clib.compat.bbclib.MsgType</i> attribute), 9	RESPONSE_ECDH_KEY_EXCHANGE ( <i>bb-clib.libs.bbclib_msgtype.MsgType</i> attribute), 18
REQUEST_SEARCH_WITH_CONDITIONS ( <i>bb-clib.libs.bbclib_msgtype.MsgType</i> attribute), 17	RESPONSE_GATHER_SIGNATURE ( <i>bb-clib.compat.bbclib.MsgType</i> attribute), 10
REQUEST_SET_STATIC_NODE ( <i>bb-clib.compat.bbclib.MsgType</i> attribute), 9	RESPONSE_GATHER_SIGNATURE ( <i>bb-clib.libs.bbclib_msgtype.MsgType</i> attribute), 18
REQUEST_SET_STATIC_NODE ( <i>bb-clib.libs.bbclib_msgtype.MsgType</i> attribute), 17	RESPONSE_GET_CONFIG ( <i>bb-clib.compat.bbclib.MsgType</i> attribute), 10
REQUEST_SETUP_DOMAIN ( <i>bb-clib.compat.bbclib.MsgType</i> attribute), 9	RESPONSE_GET_CONFIG ( <i>bb-clib.libs.bbclib_msgtype.MsgType</i> attribute), 18
REQUEST_SETUP_DOMAIN ( <i>bb-clib.libs.bbclib_msgtype.MsgType</i> attribute), 17	RESPONSE_GET_DOMAINLIST ( <i>bb-clib.compat.bbclib.MsgType</i> attribute), 10
REQUEST_SIGNATURE ( <i>bbclib.compat.bbclib.MsgType</i> attribute), 10	RESPONSE_GET_DOMAINLIST ( <i>bb-clib.libs.bbclib_msgtype.MsgType</i> attribute), 18
REQUEST_SIGNATURE ( <i>bb-clib.libs.bbclib_msgtype.MsgType</i> attribute), 17	RESPONSE_GET_FORWARDING_LIST ( <i>bb-clib.compat.bbclib.MsgType</i> attribute), 10
REQUEST_TRAVERSE_TRANSACTIONS ( <i>bb-clib.compat.bbclib.MsgType</i> attribute), 10	RESPONSE_GET_FORWARDING_LIST ( <i>bb-clib.libs.bbclib_msgtype.MsgType</i> attribute), 18
REQUEST_TRAVERSE_TRANSACTIONS ( <i>bb-clib.libs.bbclib_msgtype.MsgType</i> attribute), 17	RESPONSE_GET_NEIGHBORLIST ( <i>bb-clib.compat.bbclib.MsgType</i> attribute), 10
REQUEST_VERIFY_HASH_IN_SUBSYS ( <i>bb-clib.compat.bbclib.MsgType</i> attribute), 10	RESPONSE_GET_NEIGHBORLIST ( <i>bb-clib.libs.bbclib_msgtype.MsgType</i> attribute), 18
REQUEST_VERIFY_HASH_IN_SUBSYS ( <i>bb-clib.libs.bbclib_msgtype.MsgType</i> attribute), 17	RESPONSE_GET_NODEID ( <i>bb-clib.compat.bbclib.MsgType</i> attribute), 10
reset_error() (in module <i>bbclib.compat.bbclib</i> ), 12	RESPONSE_GET_NODEID ( <i>bb-clib.libs.bbclib_msgtype.MsgType</i> attribute), 18
RESPONSE_CLOSE_DOMAIN ( <i>bb-clib.compat.bbclib.MsgType</i> attribute), 10	RESPONSE_GET_NOTIFICATION_LIST ( <i>bb-clib.compat.bbclib.MsgType</i> attribute), 10
RESPONSE_CLOSE_DOMAIN ( <i>bb-clib.libs.bbclib_msgtype.MsgType</i> attribute), 17	RESPONSE_GET_NOTIFICATION_LIST ( <i>bb-clib.libs.bbclib_msgtype.MsgType</i> attribute), 18
RESPONSE_COUNT_TRANSACTIONS ( <i>bb-clib.compat.bbclib.MsgType</i> attribute), 10	RESPONSE_GET_STATS ( <i>bb-clib.compat.bbclib.MsgType</i> attribute), 10
RESPONSE_COUNT_TRANSACTIONS ( <i>bb-clib.libs.bbclib_msgtype.MsgType</i> attribute), 17	RESPONSE_GET_STATS ( <i>bb-clib.libs.bbclib_msgtype.MsgType</i> attribute), 18
RESPONSE_CROSS_REF_LIST ( <i>bb-clib.compat.bbclib.MsgType</i> attribute), 10	RESPONSE_GET_USERS ( <i>bb-clib.compat.bbclib.MsgType</i> attribute), 10
RESPONSE_CROSS_REF_LIST ( <i>bb-clib.libs.bbclib_msgtype.MsgType</i> attribute), 17	RESPONSE_GET_USERS ( <i>bb-clib.libs.bbclib_msgtype.MsgType</i> attribute), 18
RESPONSE_CROSS_REF_VERIFY ( <i>bb-</i>	



RESPONSE\_INSERT (*bbclib.compat.bbclib.MsgType attribute*), 10  
 RESPONSE\_INSERT (*bbclib.libs.bbclib\_msgtype.MsgType attribute*), 18  
 RESPONSE\_MANIP\_LEDGER\_SUBSYS (*bbclib.compat.bbclib.MsgType attribute*), 10  
 RESPONSE\_MANIP\_LEDGER\_SUBSYS (*bbclib.libs.bbclib\_msgtype.MsgType attribute*), 18  
 RESPONSE\_REGISTER\_HASH\_IN\_SUBSYS (*bbclib.compat.bbclib.MsgType attribute*), 10  
 RESPONSE\_REGISTER\_HASH\_IN\_SUBSYS (*bbclib.libs.bbclib\_msgtype.MsgType attribute*), 18  
 RESPONSE\_SEARCH\_TRANSACTION (*bbclib.compat.bbclib.MsgType attribute*), 10  
 RESPONSE\_SEARCH\_TRANSACTION (*bbclib.libs.bbclib\_msgtype.MsgType attribute*), 18  
 RESPONSE\_SEARCH\_WITH\_CONDITIONS (*bbclib.compat.bbclib.MsgType attribute*), 10  
 RESPONSE\_SEARCH\_WITH\_CONDITIONS (*bbclib.libs.bbclib\_msgtype.MsgType attribute*), 18  
 RESPONSE\_SET\_STATIC\_NODE (*bbclib.compat.bbclib.MsgType attribute*), 10  
 RESPONSE\_SET\_STATIC\_NODE (*bbclib.libs.bbclib\_msgtype.MsgType attribute*), 18  
 RESPONSE\_SETUP\_DOMAIN (*bbclib.compat.bbclib.MsgType attribute*), 10  
 RESPONSE\_SETUP\_DOMAIN (*bbclib.libs.bbclib\_msgtype.MsgType attribute*), 18  
 RESPONSE\_SIGNATURE (*bbclib.compat.bbclib.MsgType attribute*), 10  
 RESPONSE\_SIGNATURE (*bbclib.libs.bbclib\_msgtype.MsgType attribute*), 18  
 RESPONSE\_TRAVERSE\_TRANSACTIONS (*bbclib.compat.bbclib.MsgType attribute*), 10  
 RESPONSE\_TRAVERSE\_TRANSACTIONS (*bbclib.libs.bbclib\_msgtype.MsgType attribute*), 18  
 RESPONSE\_VERIFY\_HASH\_IN\_SUBSYS (*bbclib.compat.bbclib.MsgType attribute*), 10  
 RESPONSE\_VERIFY\_HASH\_IN\_SUBSYS (*bbclib.libs.bbclib\_msgtype.MsgType attribute*), 18  
 S  
 serialize() (*bbclib.compat.bbclib.BBcAsset method*), 2  
 serialize() (*bbclib.compat.bbclib.BBcCrossRef method*), 2  
 serialize() (*bbclib.compat.bbclib.BBcEvent method*), 3  
 serialize() (*bbclib.compat.bbclib.BBcPointer method*), 4  
 serialize() (*bbclib.compat.bbclib.BBcReference method*), 5  
 serialize() (*bbclib.compat.bbclib.BBcRelation method*), 5  
 serialize() (*bbclib.compat.bbclib.BBcSignature method*), 6  
 serialize() (*bbclib.compat.bbclib.BBcTransaction method*), 7  
 serialize() (*bbclib.compat.bbclib.BBcWitness method*), 8  
 serialize() (*in module bbclib*), 26  
 serialize\_obj() (*bbclib.compat.bbclib.BBcTransaction method*), 7  
 set\_error() (*in module bbclib.compat.bbclib*), 12  
 set\_format\_type() (*bbclib.compat.bbclib.BBcTransaction method*), 7  
 set\_sig\_index() (*bbclib.libs.bbclib\_transaction.BBcTransaction method*), 22  
 sign() (*bbclib.compat.bbclib.BBcTransaction method*), 7  
 sign() (*bbclib.compat.bbclib.KeyPair method*), 8  
 sign() (*bbclib.libs.bbclib\_keypair.KeyPair method*), 16  
 sign() (*bbclib.libs.bbclib\_transaction.BBcTransaction method*), 22  
 str\_binary() (*in module bbclib.compat.bbclib*), 12  
 str\_binary() (*in module bbclib.libs.bbclib\_utils*), 24  
 strip() (*bbclib.libs.bbclib\_wire.BBcFormat class method*), 25  
 T  
 to\_1byte() (*in module bbclib.compat.bbclib*), 12  
 to\_1byte() (*in module bbclib.libs.bbclib\_utils*), 24  
 to\_2byte() (*in module bbclib.compat.bbclib*), 12  
 to\_2byte() (*in module bbclib.libs.bbclib\_utils*), 24  
 to\_4byte() (*in module bbclib.compat.bbclib*), 12  
 to\_4byte() (*in module bbclib.libs.bbclib\_utils*), 24  
 to\_8byte() (*in module bbclib.compat.bbclib*), 12  
 to\_8byte() (*in module bbclib.libs.bbclib\_utils*), 24  
 to\_bigint() (*in module bbclib.compat.bbclib*), 12  
 to\_bigint() (*in module bbclib.libs.bbclib\_utils*), 24  
 to\_binary() (*bbclib.compat.bbclib.KeyPair method*), 8  
 to\_binary() (*bbclib.libs.bbclib\_keypair.KeyPair method*), 16

## U

`unpack()` (*bbclib.libs.bbclib\_asset.BBcAsset* method), 13

`unpack()` (*bbclib.libs.bbclib\_crossref.BBcCrossRef* method), 14

`unpack()` (*bbclib.libs.bbclib\_event.BBcEvent* method), 15

`unpack()` (*bbclib.libs.bbclib\_pointer.BBcPointer* method), 19

`unpack()` (*bbclib.libs.bbclib\_reference.BBcReference* method), 19

`unpack()` (*bbclib.libs.bbclib\_relation.BBcRelation* method), 20

`unpack()` (*bbclib.libs.bbclib\_signature.BBcSignature* method), 20

`unpack()` (*bbclib.libs.bbclib\_transaction.BBcTransaction* method), 22

`unpack()` (*bbclib.libs.bbclib\_witness.BBcWitness* method), 26

`UNREGISTER` (*bbclib.compat.bbclib.MsgType* attribute), 10

`UNREGISTER` (*bbclib.libs.bbclib\_msgtype.MsgType* attribute), 18

## V

`validate_transaction_object()` (in module *bbclib.compat.bbclib*), 12

`validate_transaction_object()` (in module *bbclib.libs.bbclib\_utils*), 24

`verify()` (*bbclib.compat.bbclib.BBcSignature* method), 6

`verify()` (*bbclib.compat.bbclib.KeyPair* method), 8

`verify()` (*bbclib.libs.bbclib\_keypair.KeyPair* method), 16

`verify()` (*bbclib.libs.bbclib\_signature.BBcSignature* method), 21

`verify_using_cross_ref()` (in module *bbclib.compat.bbclib*), 12

`verify_using_cross_ref()` (in module *bbclib.libs.bbclib\_utils*), 24

## W

`WITH_WIRE` (*bbclib.compat.bbclib.BBcTransaction* attribute), 6

`WITH_WIRE` (*bbclib.libs.bbclib\_transaction.BBcTransaction* attribute), 21