py-bbclib Documentation

Release 1.5

beyond-blockchain.org

Oct 26, 2019

Contents:

CHAPTER 1

bbclib package

1.1 Subpackages

1.1.1 bbclib.compat package

1.1.1.1 Submodules

bbclib.compat.bbclib module

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*, *option_approver_num_numerator=0*, *option_approver_num_denominator=0*, *option_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*, *format_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*, *format_type=0*, *id_length=32*)
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 rivate key in this object

class bbclib.compat.bbclib.**KeyPairFast**(*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 rivate 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*, *asset_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*, *asset_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*, *format_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

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*, *asset_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_asset_hash 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.libs.bbclib_asset_hash.**BBcAssetHash**(*asset_ids=None*, *id_length=None*)
Bases: object

AssetHash part in a transaction

add(*asset_ids=None*)
Add parts in this object

digest()
Return digest

The digest is the first entry of asset_ids

Returns asset_id

Return type bytes

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_asset_raw module

class bbclib.libs.bbclib_asset_raw.**BBcAssetRaw**(*asset_id=None*, *asset_body=None*, *id_length=None*)
Bases: object

AssetRaw part in a transaction

In this object, asset_id should be given externally, meaning that this object does not care about how to calculate the digest.

add(*asset_id=None*, *asset_body=None*)
Add parts in this object

digest()
Return digest

The digest corresponds to the asset_id of this object. The asset_id is given externally, so bbclib does not care about how to calculate the digest of the asset_body.

Returns asset_id

Return type bytes

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_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*, *option_approver_num_numerator=0*, *option_approver_num_denominator=0*, *option_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.

bbclib.libs.bbclib_keypair.**KeyPair**
alias of *bbclib.libs.bbclib_keypair.KeyPairPy*

class bbclib.libs.bbclib_keypair.**KeyPairPy**(*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(*privkey*)
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_keypair_fast module

bbclib.libs.bbclib_msgtype module

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

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

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

class bbclib.libs.bbclib_relation.**BBcRelation**(*asset_group_id=None*, *id_length=None*, *version=1*)
Bases: object

Relation part in a transaction

add(*asset_group_id=None*, *asset=None*, *asset_raw=None*, *asset_hash=None*, *pointer=None*)
Add parts

pack()
Pack this object

Returns packed binary data

Return type bytes

unpack(*data*, *version=2*)
Unpack data into transaction object

Parameters

- **data** (*bytes*) – packed binary data
- **version** (*int*) – version of the data format

Returns True if successful

Return type bool

bbclib.libs.bbclib_signature module

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

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

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

Parameters

- **relation** (BBcRelation) – BBcRelation object to manipulate
- **ref_transaction_id** (*bytes*) – transaction_id of the transaction that the base transaction object refers to
- **ref_asset_id** (*bytes*) – asset_id of the asset that the transaction object refers to

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

Parameters

- **transaction** (BBcTransaction) – the base transaction object to manipulate
- **asset_group_id** (*bytes*) – asset_group_id of the asset in the object
- **ref_transaction_obj** (BBcTransaction) – the transaction object that the base transaction object refers to
- **event_index_in_ref** (*int*) – the number of BBcEvent object to include in the transaction that the base transaction object refers to

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

Parameters

- **transaction** (BBcTransaction) – transaction object to manipulate
- **relation_idx** (*int*) – the number of BBcRelation object to include in the transaction
- **asset_group_id** (*bytes*) – asset_group_id of the asset in the object
- **user_id** (*bytes*) – user_id of the owner of the asset
- **asset_body** (*str|bytes|dict*) – asset data
- **asset_file** (*bytes*) – file data (binary) for asset

bbclib.libs.bbclib_utils.**add_relation_asset_hash**(*transaction*, *relation_idx*, *asset_group_id*, *asset_ids=None*)

Utility to add BBcRelation object with BBcAssetHash in the transaction

Parameters

- **transaction** (BBcTransaction) – transaction object to manipulate
- **relation_idx** (*int*) – the number of BBcRelation object to include in the transaction
- **asset_group_id** (*bytes*) – asset_group_id of the asset in the object
- **asset_ids** (*list(bytes)*) – list of the identifiers of assets

bbclib.libs.bbclib_utils.**add_relation_asset_raw**(*transaction*, *relation_idx*, *asset_group_id*, *asset_id=None*, *asset_body=None*)

Utility to add BBcRelation object with BBcAssetRaw in the transaction

Parameters

- **transaction** (BBcTransaction) – transaction object to manipulate
- **relation_idx** (*int*) – the number of BBcRelation object to include in the transaction
- **asset_group_id** (*bytes*) – asset_group_id of the asset in the object
- **asset_id** (*bytes*) – the identifier of the asset
- **asset_body** (*str|bytes|dict*) – asset data

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

Parameters

- **transaction** (BBcTransaction) – the base transaction object to manipulate
- **relation_idx** (*int*) – the number of BBcRelation object to include in the transaction
- **ref_transaction_id** (*bytes*) – transaction_id of the transaction that the base transaction object refers to
- **ref_asset_id** (*bytes*) – asset_id of the asset that the transaction object refers to

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

bbclib.libs.bbclib_utils.**convert_id_to_string**(*data*, *bytelen=32*)
Convert binary data to hex string

Parameters

- **data** (*bytes*) – data to convert
- **bytelen** (*int*) – length of the result

Returns converted string

Return type str

bbclib.libs.bbclib_utils.**convert_idstring_to_bytes**(*datastr*, *bytelen=32*)
Convert hex string to binary data

Parameters

- **datastr** (*str*) – data to convert
- **bytelen** (*int*) – length of the result

Returns converted byte data

Return type bytes

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 random bytes

Parameters **length** (*int*) – length of the result

Returns random bytes

Return type bytes

bbclib.libs.bbclib_utils.**make_relation_with_asset**(*asset_group_id*, *user_id*, *asset_body=None*, *asset_file=None*)
Utility to make BBcRelation object with BBcAsset

Parameters

- **asset_group_id** (*bytes*) – asset_group_id of the asset in the object
- **user_id** (*bytes*) – user_id of the owner of the asset
- **asset_body** (*str|bytes|dict*) – asset data
- **asset_file** (*bytes*) – file data (binary) for asset

Returns created BBcRelation object

Return type *BBcRelation*

bbclib.libs.bbclib_utils.**make_relation_with_asset_hash**(*asset_group_id*, *asset_ids=None*)

Utility to make BBcRelation object with BBcAssetHash

Parameters

- **asset_group_id** (*bytes*) – asset_group_id of the asset in the object
- **asset_ids** (*list(bytes)*) – list of the identifiers of assets

Returns created BBcRelation object

Return type *BBcRelation*

bbclib.libs.bbclib_utils.**make_relation_with_asset_raw**(*asset_group_id*, *asset_id=None*, *asset_body=None*)

Utility to make BBcRelation object with BBcAssetRaw

Parameters

- **asset_group_id** (*bytes*) – asset_group_id of the asset in the object
- **asset_id** (*bytes*) – the identifier of the asset
- **asset_body** (*str|bytes|dict*) – asset data

Returns created BBcRelation object

Return type *BBcRelation*

bbclib.libs.bbclib_utils.**make_transaction**(*event_num=0*, *relation_num=0*, *witness=False*, *version=1*)

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
- **version** (*int*) – version of the transaction format

Returns

Return type *BBcTransaction*

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

Unpack signature data

Parameters **data** (*bytes*) – Serialized data of BBcSignature object

Returns BBcSignature object

Return type *BBcSignature*

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*, *sigdata*)
Confirm the existence of the transaction using cross_ref

Parameters

- **domain_id** (*bytes*) – target domain_id
- **transaction_id** (*bytes*) – target transaction_id of which existence you want to confirm
- **transaction_base_digest** (*bytes*) – digest obtained from the outer domain
- **cross_ref_data** (*bytes*) – packed BBcCrossRef object
- **sigdata** (*bytes*) – packed signature

Returns True if valid

Return type bool

bbclib.libs.bbclib_wire module

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

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

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

Python Module Index

b

Index

C

D

E

F

G

I

K

M

N

P

R

S

T

U

V

W