py-factom-did Documentation

Release 0.5.0

Peter Asenov, Valentin Ganev

Nov 04, 2019

Contents

CHAPTER 1

Submodules

1.1 client.deactivator module

class factom_did.client.deactivator.**DIDDeactivator**(*did*)
Bases: object

Facilitates the creation of a DIDDeactivation entry.

did
The DID object to update

Type *client.did.DID*

export_entry_data()
Constructs a signed DIDDeactivation entry ready for recording on-chain.

Returns A dictionary with ExtIDs and content for the entry

Return type dict

record_on_chain(*factomd*, *walletd*, *ec_address*, *verbose=False*)
Attempts to record the DIDDeactivation entry on-chain.

Parameters

- **factomd** (*obj*) – Factomd instance, instantiated from the Python factom-api package.
- **walletd** (*obj*) – Factom walletd instance, instantiated from the Python factom-api package.
- **ec_address** (*str*) – EC address used to pay for the chain & entry creation.
- **verbose** (*bool, optional*) – If true, display the contents of the entry that will be recorded on-chain.

Raises RuntimeError – If the entry cannot be recorded

1.2 client.did module

class factom_did.client.did.**DID**(*did=None*, *management_keys=None*, *did_keys=None*, *services=None*, *spec_version='0.2.0'*)

Bases: object

Enables the construction of a DID document, by facilitating the construction of management keys and DID keys and the addition of services. Allows exporting of the resulting DID object into a format suitable for recording on the Factom blockchain.

Provides encryption functionality of private keys for the DID and their export to a string or to a JSON file.

did
The decentralized identifier, a 32 byte hexadecimal string

Type str, optional

management_keys
A list of management keys

Type *ManagementKey*[], optional

did_keys
A list of DID keys

Type *DIDKey*[], optional

services
A list of services

Type *Service*[], optional

deactivate()

Raises RuntimeError – If no management keys are available for the DID

Returns

Return type *DIDDeactivator*

did_key(*alias*, *purpose*, *key_type=<KeyType.EdDSA: 'Ed25519VerificationKey'>*, *controller=None*, *priority_requirement=None*)
Creates a new DID key for the DID.

Parameters

- **alias** (*str*) – A human-readable nickname for the key. It should be unique across the keys defined in the DID document.
- **purpose** (DIDKeyPurpose *or* DIDKeyPurpose*[]*) – Shows what purpose(s) the key serves. (PublicKey, AuthenticationKey or both)
- **key_type** (KeyType, *optional*) – Identifies the type of signature that the key pair can be used to generate and verify.
- **controller** (*str, optional*) – An entity that will be making the signatures. It must be a valid DID. If the argument is not passed in, the default value is used which is the current DID itself.
- **priority_requirement** (*int, optional*) – A non-negative integer showing the minimum hierarchical level a key must have in order to remove this key.

export_encrypted_keys_as_json(*password*)
Exports encrypted keys as JSON.

Parameters **password** (*str*) – A password to use for the encryption of the keys.

Returns Encrypted keys JSON.

Return type str

export_encrypted_keys_as_str(*password*)
Exports encrypted keys cipher text.

Parameters **password** (*str*) – A password to use for the encryption of the keys.

Returns Encrypted keys cipher text.

Return type str

export_entry_data()
Exports content that can be recorded on-chain to create the DID.

Returns A dictionary with the ExtIDs and entry content of strings used that are the header columns.

Return type dict

Raises ValueError – If there are no management keys. If there is no management key with priority 0. If the entry size exceeds the entry size limit.

get_chain()

Returns The chain ID where this DID is (or will be) stored

Return type str

id

static is_valid_did(*did*)

mainnet()
Sets the DID network to mainnet.

management_key(*alias*, *priority*, *key_type=<KeyType.EdDSA: 'Ed25519VerificationKey'>*, *controller=None*, *priority_requirement=None*)
Creates a new management key for the DID.

Parameters

- **alias** (*str*) – A human-readable nickname for the key. It should be unique across the keys defined in the DID document.
- **priority** (*int*) – A non-negative integer showing the hierarchical level of the key. Keys with lower priority override keys with higher priority.
- **key_type** (KeyType, *optional*) – Identifies the type of signature that the key pair can be used to generate and verify.
- **controller** (*str, optional*) – An entity that controls the key. It must be a valid DID. If the argument is not passed in, the default value is used which is the current DID itself.
- **priority_requirement** (*int, optional*) – A non-negative integer showing the minimum hierarchical level a key must have in order to remove this key.

method_spec_version_upgrade(*new_spec_version*)

Parameters **new_spec_version** (*str*) – The new DID Method version

Raises

- RuntimeError – If no management keys are available for the DID

- `ValueError` – If the new version is not an upgrade on the current version

Returns

Return type *DIDVersionUpgrader*

record_on_chain (*factomd*, *walletd*, *ec_address*, *verbose=False*)
Attempts to create the DIDManagement chain.

Parameters

- **factomd** (*obj*) – Factomd instance, instantiated from the Python factom-api package.
- **walletd** (*obj*) – Factom walletd instance, instantiated from the Python factom-api package.
- **ec_address** (*str*) – EC address used to pay for the chain & entry creation.
- **verbose** (*bool, optional*) – If true, display the contents of the entry that will be recorded on-chain.

Raises `RuntimeError` – If the chain cannot be created

service (*alias*, *service_type*, *endpoint*, *priority_requirement=None*)
Adds a new service to the DID Document.

Parameters

- **alias** (*str*) – A human-readable nickname for the service endpoint. It should be unique across the services defined in the DID document.
- **service_type** (*str*) – Type of the service endpoint.
- **endpoint** (*str*) – A service endpoint may represent any type of service the subject wishes to advertise, including decentralized identity management services for further discovery, authentication, authorization, or interaction. The service endpoint must be a valid URL.
- **priority_requirement** (*int, optional*) – A non-negative integer showing the minimum hierarchical level a key must have in order to remove this service.

testnet ()
Sets the DID network to testnet.

update ()

Raises `RuntimeError` – If no management keys are available for the DID

Returns An object allowing updates to the existing DID

Return type *DIDUpdater*

1.3 client.encryptor module

factom_did.client.encryptor.**encrypt_keys** (*management_keys*, *did_keys*, *password*)
Encrypts keys with a password.

Parameters

- **management_keys** (*ManagementKeyModel[]*) – A list of management keys to be encrypted.
- **did_keys** (*DidKeyModel[]*) – A list of did keys to be encrypted.

- **password** (*str*) – A password to use for the encryption of the keys.

Returns An object containing salt, initial vector, tag and encrypted data.

Return type obj

factom_did.client.encryptor.**decrypt_keys_from_str**(*cipher_text_b64*, *password*, *encryption_algo='AES-GCM'*)

Decrypts keys from cipher text and password.

Parameters

- **cipher_text_b64** (*str*) – Base 64 encoded cipher text.
- **password** (*str*) – A password used for the encryption of the keys.
- **encryption_algo** (*str*) – The encryption algorithm used. Currently only 'AES-GCM' is supported

Returns An object containing dictionaries of decrypted management and did keys.

Return type obj

Raises ValueError – If the cipher text or the password used for the encryption is invalid.

factom_did.client.encryptor.**decrypt_keys_from_json_str**(*encrypted_keys_json_str*, *password*)

Decrypts keys from JSON string and password. The JSON string must have a schema compatible with the one produced by DID.export_encrypted_keys_as_json():

'{

"encryptionAlgo": { "salt": ..., "iv": ..., "name": ..., "tagLength": ...,

}, "data": ... (encrypted private keys), "did": ...

}'

Parameters

- **encrypted_keys_json_str** (*str*) – JSON string containing encrypted keys data.
- **password** (*str*) – A password used for the encryption of the keys.

Returns An object containing dictionaries of decrypted management and did keys.

Return type obj

Raises ValueError – If the JSON or the password used for the encryption is invalid.

factom_did.client.encryptor.**decrypt_keys_from_json_file**(*file_path*, *password*)

Decrypts keys from JSON file and password. The file must contain valid JSON with a schema compatible with the one produced by DID.export_encrypted_keys_as_json(). See decrypt_keys_from_json_str for details.

Parameters

- **file_path** (*str*) – Path to a file to read from.
- **password** (*str*) – A password used for the encryption of the keys.

Returns An object containing dictionaries of decrypted management and did keys.

Return type obj

Raises ValueError – If the file or the password is invalid.

1.4 client.enums module

class factom_did.client.enums.**KeyType**
Bases: enum.Enum

An enumeration.

ECDSA = 'ECDSASecp256k1VerificationKey'

EdDSA = 'Ed25519VerificationKey'

RSA = 'RSAVerificationKey'

from_str = <function KeyType.from_str>

class factom_did.client.enums.**EntryType**
Bases: enum.Enum

An enumeration.

Create = 'DIDManagement'

Deactivation = 'DIDDeactivation'

Update = 'DIDUpdate'

VersionUpgrade = 'DIDMethodVersionUpgrade'

class factom_did.client.enums.**DIDKeyPurpose**
Bases: enum.Enum

An enumeration.

AuthenticationKey = 'authentication'

PublicKey = 'publicKey'

from_str = <function DIDKeyPurpose.from_str>

class factom_did.client.enums.**Network**
Bases: enum.Enum

An enumeration.

Mainnet = 'mainnet'

Testnet = 'testnet'

Unspecified = ''

from_str = <function Network.from_str>

1.5 client.keys package

1.5.1 Submodules

client.keys.abstract module

class factom_did.client.keys.abstract.**AbstractDIDKey**(*alias*, *key_type*, *controller*, *priority_requirement*, *public_key=None*, *private_key=None*)

Bases: `object`

Represents the common fields and functionality in a ManagementKey and a DidKey.

alias
A human-readable nickname for the key.

Type str

key_type
Identifies the type of signature that the key pair can be used to generate and verify.

Type *KeyType*

controller
An entity that controls the key.

Type str

priority_requirement
A non-negative integer showing the minimum hierarchical level a key must have in order to remove this key.

Type int

public_key
The public key.

Type bytes or str, optional

private_key
The private key.

Type bytes or str, optional

static from_entry_dict(*entry_dict*, *version='1.0.0'*)
Creates an AbstractDIDKey object from an on-chain entry

Parameters

- **entry_dict** (*dict*) – The on-chain entry, represented as a Python dictionary
- **version** (*str*) – The entry schema version

Returns

Return type *AbstractDIDKey*

Raises `NotImplementedError` – If the supplied version is not supported

full_id(*did*)
Constructs the full ID of the key.

Parameters **did** (*str*) –

Returns The full id for the key, constituting of the DID_METHOD_NAME, the network, the chain ID and the key alias.

Return type str

private_key

public_key

rotate()
Generates new key pair for the key.

sign(*message*, *hash_f=None*)

signing_key

to_entry_dict(*did*, *version='1.0.0'*)
Converts the object to a dictionary suitable for recording on-chain.

Parameters

- **did** (*str*) – The DID with which this key is associated. Note that this can be different from the key controller.
- **version** (*str*) – The entry schema version

Returns Dictionary with *id*, *type*, *controller* and an optional *priorityRequirement* fields. In addition to those, there is one extra field for the public key: if the selected signature type is SignatureType.RSA, then this field is called *publicKeyPem*, otherwise it is called *publicKeyBase58*.

Return type dict

verify(*message*, *signature*, *hash_f=None*)

verifying_key

client.keys.did module

class factom_did.client.keys.did.**DIDKey**(*alias*, *purpose*, *key_type*, *controller*, *priority_requirement=None*, *public_key=None*, *private_key=None*)
Bases: *factom_did.client.keys.abstract.AbstractDIDKey*

Application-level key, which can be used for authentication, signing requests, encryption, decryption, etc.

alias

Type str

purpose
Shows what purpose(s) the key serves. (PublicKey, AuthenticationKey or both)

Type *DIDKeyPurpose* or *DIDKeyPurpose*[]

key_type

Type *KeyType*

controller

Type str

priority_requirement

Type int, optional

public_key

Type str, optional

private_key

Type str, optional

static from_entry_dict(*entry_dict*, *version='1.0.0'*)
Creates an AbstractDIDKey object from an on-chain entry

Parameters

- **entry_dict** (*dict*) – The on-chain entry, represented as a Python dictionary
- **version** (*str*) – The entry schema version

Returns

Return type *AbstractDIDKey*

Raises NotImplementedError – If the supplied version is not supported

to_entry_dict(*did*, *version='1.0.0'*)
Converts the object to a dictionary suitable for recording on-chain.

Parameters

- **did** (*str*) – The DID with which this key is associated. Note that this can be different from the key controller.
- **version** (*str*) – The entry schema version

Returns Dictionary with *id*, *type*, *controller* and an optional *priorityRequirement* fields. In addition to those, there is one extra field for the public key: if the selected signature type is SignatureType.RSA, then this field is called *publicKeyPem*, otherwise it is called *publicKeyBase58*.

Return type dict

client.keys.ecdsa module

class factom_did.client.keys.ecdsa.**ECDSASecp256k1Key**(*public_key=None*, *private_key=None*)

Bases: object

Representation of an ECDSASecp256k1 key. Instances of this class allow signing of messages and signature verification, as well as key creation and derivation of a public key from a private key.

ON_CHAIN_PUB_KEY_NAME = 'publicKeyBase58'

get_public_key_on_chain_repr()

private_key

public_key

sign(*message*, *hash_f=<built-in function openssl_sha256>*)
Signs a message with the existing private key and signature type.

The message is hashed before being signed, with the provided hash function. The default hash function used is SHA-256.

Parameters

- **message** (*bytes*) – The message to sign.
- **hash_f** (*function, optional*) – The hash function used to compute the digest of the message before signing it.

Returns The bytes of the signatures.

Return type bytes

Raises `AssertionError` – If the supplied message is not bytes, or if a private key has not been specified.

verify(*message*, *signature*, *hash_f=<built-in function openssl_sha256>*)
Verifies the signature of the given message

Parameters

- **message** (*bytes*) – The (allegedly) signed message.
- **signature** (*bytes*) – The signature to verify.
- **hash_f** (*function, optional*) – The hash function used to compute the digest of the message.

Returns True if the signature is successfully verified, False otherwise.

Return type bool

client.keys.eddsa module

class factom_did.client.keys.eddsa.**Ed25519Key**(*public_key=None*, *private_key=None*)
Bases: object

Representation of an Ed25519 key. Instances of this class allow signing of messages and signature verification, as well as key creation and derivation of a public key from a private key.

ON_CHAIN_PUB_KEY_NAME = 'publicKeyBase58'

get_public_key_on_chain_repr()

private_key

public_key

sign(*message*, *hash_f=<built-in function openssl_sha256>*)
Signs a message with the existing private key and signature type.

The message is hashed before being signed, with the provided hash function. The default hash function used is SHA-256.

Parameters

- **message** (*bytes*) – The message to sign.
- **hash_f** (*function, optional*) – The hash function used to compute the digest of the message before signing it.

Returns The bytes of the signatures.

Return type bytes

Raises `AssertionError` – If the supplied message is not bytes, or if a private key has not been specified.

verify(*message*, *signature*, *hash_f=<built-in function openssl_sha256>*)
Verifies the signature of the given message

Parameters

- **message** (*bytes*) – The (allegedly) signed message.
- **signature** (*bytes*) – The signature to verify.

- **hash_f** (*function, optional*) – The hash function used to compute the digest of the message.

Returns True if the signature is successfully verified, False otherwise.

Return type bool

client.keys.management module

class factom_did.client.keys.management.**ManagementKey**(*alias*, *priority*, *key_type*, *controller*, *priority_requirement=None*, *public_key=None*, *private_key=None*)

Bases: *factom_did.client.keys.abstract.AbstractDIDKey*

A key used to sign updates for an existing DID.

alias

Type str

priority

A non-negative integer showing the hierarchical level of the key. Keys with lower priority override keys with higher priority.

Type int

key_type

Type *KeyType*

controller

Type str

priority_requirement

Type int, optional

public_key

Type str, optional

private_key

Type str, optional

static from_entry_dict(*entry_dict*, *version='1.0.0'*)

Creates an AbstractDIDKey object from an on-chain entry

Parameters

- **entry_dict** (*dict*) – The on-chain entry, represented as a Python dictionary
- **version** (*str*) – The entry schema version

Returns

Return type *AbstractDIDKey*

Raises NotImplementedError – If the supplied version is not supported

to_entry_dict(*did*, *version='1.0.0'*)

Converts the object to a dictionary suitable for recording on-chain.

Parameters

- **did** (*str*) – The DID with which this key is associated. Note that this can be different from the key controller.
- **version** (*str*) – The entry schema version

Returns Dictionary with *id*, *type*, *controller* and an optional *priorityRequirement* fields. In addition to those, there is one extra field for the public key: if the selected signature type is SignatureType.RSA, then this field is called *publicKeyPem*, otherwise it is called *publicKeyBase58*.

Return type dict

client.keys.rsa module

class factom_did.client.keys.rsa.**RSAKey**(*public_key=None*, *private_key=None*)
Bases: object

Representation of an RSA key. Instances of this class allow signing of messages and signature verification, as well as key creation and derivation of a public key from a private key.

ON_CHAIN_PUB_KEY_NAME = 'publicKeyPem'

get_public_key_on_chain_repr()

private_key

public_key

sign(*message*, *hash_f=<function new>*)
Signs a message with the existing private key and signature type.

The message is hashed before being signed, with the provided hash function. The default hash function used is SHA-256.

Parameters

- **message** (*bytes*) – The message to sign.
- **hash_f** (*function, optional*) – The hash function used to compute the digest of the message before signing it.

Returns The bytes of the signatures.

Return type bytes

Raises AssertionError – If the supplied message is not bytes, or if a private key has not been specified.

verify(*message*, *signature*, *hash_f=<function new>*)
Verifies the signature of the given message

Parameters

- **message** (*bytes*) – The (allegedly) signed message.
- **signature** (*bytes*) – The signature to verify.
- **hash_f** (*function, optional*) – The hash function used to compute the digest of the message.

Returns True if the signature is successfully verified, False otherwise.

Return type bool

1.6 client.service module

class factom_did.client.service.**Service**(*alias*, *service_type*, *endpoint*, *priority_requirement=None*)

Bases: object

Represent a service associated with a DID. A service is an end-point, which can be used to communicate with the DID or to carry out different tasks on behalf of the DID (such as signatures, e.g.)

alias
A human-readable nickname for the service endpoint.

Type str

service_type
Type of the service endpoint (e.g. email, credential store).

Type str

endpoint
A service endpoint may represent any type of service the subject wishes to advertise, including decentralized identity management services for further discovery, authentication, authorization, or interaction. The service endpoint must be a valid URL.

Type str

priority_requirement
A non-negative integer showing the minimum hierarchical level a key must have in order to remove this service.

Type int, optional

static from_entry_dict(*entry_dict*, *version='1.0.0'*)

full_id(*did*)

Returns The full id for the service, constituting of the DID_METHOD_NAME, the controller and the service alias.

Return type str

to_entry_dict(*did*, *version='1.0.0'*)
Converts the object to a dictionary suitable for recording on-chain.

Parameters

- **did** (*str*) – The DID to which this service belongs.
- **version** (*str*) – The entry schema version

Raises NotImplementedError – If the entry schema version is not supported

1.7 client.updater module

class factom_did.client.updater.**DIDUpdater**(*did*)

Bases: object

Facilitates the creation of an update entry for an existing DID.

Provides support for adding and revoking management keys, DID keys and services.

did
The DID object to update

Type *client.did.DID*

add_did_key(*alias*, *purpose*, *key_type=<KeyType.EdDSA: 'Ed25519VerificationKey'>*, *controller=None*, *priority_requirement=None*)
Adds a DID key to the DID object.

Parameters

- **alias** (*str*) –
- **purpose** (*did.enums.DIDKeyPurpose*) –
- **key_type** (KeyType, *optional*) –
- **controller** (*str, optional*) –
- **priority_requirement** (*int, optional*) –

add_management_key(*alias*, *priority*, *key_type=<KeyType.EdDSA: 'Ed25519VerificationKey'>*, *controller=None*, *priority_requirement=None*)
Adds a management key to the DID object.

Parameters

- **alias** (*str*) –
- **priority** (*int*) –
- **key_type** (KeyType, *optional*) –
- **controller** (*str, optional*) –
- **priority_requirement** (*int, optional*) –

add_service(*alias*, *service_type*, *endpoint*, *priority_requirement=None*)
Adds a service to the DID object.

Parameters

- **alias** (*str*) –
- **service_type** (*str*) –
- **endpoint** (*str*) –
- **priority_requirement** (*int, optional*) –

static exists_management_key_with_priority_zero(*active_management_keys*, *new_management_keys*, *management_keys_to_revoke*)
Checks if a management key of priority zero would be present if the management keys will be updated according to the given parameters.

Parameters

- **active_management_keys** (*set*) – The currently active management keys
- **new_management_keys** (*set*) – The management keys to be added
- **management_keys_to_revoke** (*set*) – The management keys to be revoked

Returns

Return type bool

export_entry_data()
Constructs a signed DIDUpdate entry ready for recording on-chain.

Returns A dictionary with ExtIDs and content for the entry

Return type dict

Raises RuntimeError – If a management key of sufficient priority is not available to sign the update.

get_updated()

record_on_chain(*factomd*, *walletd*, *ec_address*, *verbose=False*)
Attempts to record the DIDUpdate entry on-chain.

Parameters

- **factomd** (*obj*) – Factomd instance, instantiated from the Python factom-api package.
- **walletd** (*obj*) – Factom walletd instance, instantiated from the Python factom-api package.
- **ec_address** (*str*) – EC address used to pay for the chain & entry creation.
- **verbose** (*bool, optional*) – If true, display the contents of the entry that will be recorded on-chain.

Raises RuntimeError – If the entry cannot be recorded

revoke_did_key(*alias*)
Revokes a DID key from the DID object.

Parameters **alias** (*str*) – The alias of the key to be revoked

revoke_did_key_purpose(*alias*, *purpose*)
Revokes a single purpose of a DID key from DID object.

Parameters

- **alias** (*str*) – The alias of the DID key
- **purpose** (DIDKeyPurpose) – The purpose to revoke

revoke_management_key(*alias*)
Revokes a management key from the DID object.

Parameters **alias** (*str*) – The alias of the key to be revoked

revoke_service(*alias*)
Revokes a service from the DID object.

Parameters **alias** (*str*) – The alias of the service to be revoked

rotate_did_key(*alias*)
Rotates a DID key.

Parameters **alias** (*str*) – The alias of the DID key to be rotated

rotate_management_key(*alias*)
Rotates a management key.

Parameters **alias** (*str*) – The alias of the management key to be rotated

1.8 client.version_upgrader module

class factom_did.client.version_upgrader.**DIDVersionUpgrader**(*did*, *new_spec_version*)

Bases: object

Facilitates the creation of an DIDMethodVersionUpgrade entry for an existing DID.

did
The DID object to update

Type *client.did.DID*

new_spec_version
The new version to upgrade to

Type str

Raises ValueError – If the new version is not an upgrade on the current version

export_entry_data()
Constructs a signed DIDMethodVersionUpgrade entry ready for recording on-chain.

Returns A dictionary with ExtIDs and content for the entry

Return type dict

record_on_chain(*factomd*, *walletd*, *ec_address*, *verbose=False*)
Attempts to record the DIDMethodVersionUpgrade entry on-chain.

Parameters

- **factomd** (*obj*) – Factomd instance, instantiated from the Python factom-api package.
- **walletd** (*obj*) – Factom walletd instance, instantiated from the Python factom-api package.
- **ec_address** (*str*) – EC address used to pay for the chain & entry creation.
- **verbose** (*bool, optional*) – If true, display the contents of the entry that will be recorded on-chain.

Raises RuntimeError – If the entry cannot be recorded

CHAPTER 2

client.did module

class factom_did.client.did.**DID**(*did=None*, *management_keys=None*, *did_keys=None*, *services=None*, *spec_version='0.2.0'*)

Bases: object

Enables the construction of a DID document, by facilitating the construction of management keys and DID keys and the addition of services. Allows exporting of the resulting DID object into a format suitable for recording on the Factom blockchain.

Provides encryption functionality of private keys for the DID and their export to a string or to a JSON file.

did
The decentralized identifier, a 32 byte hexadecimal string

Type str, optional

management_keys
A list of management keys

Type *ManagementKey*[], optional

did_keys
A list of DID keys

Type *DIDKey*[], optional

services
A list of services

Type *Service*[], optional

deactivate()

Raises RuntimeError – If no management keys are available for the DID

Returns

Return type *DIDDeactivator*

did_key(*alias*, *purpose*, *key_type=<KeyType.EdDSA: 'Ed25519VerificationKey'>*, *controller=None*, *priority_requirement=None*)
Creates a new DID key for the DID.

Parameters

- **alias** (*str*) – A human-readable nickname for the key. It should be unique across the keys defined in the DID document.
- **purpose** (*DIDKeyPurpose or DIDKeyPurpose[]*) – Shows what purpose(s) the key serves. (PublicKey, AuthenticationKey or both)
- **key_type** (*KeyType, optional*) – Identifies the type of signature that the key pair can be used to generate and verify.
- **controller** (*str, optional*) – An entity that will be making the signatures. It must be a valid DID. If the argument is not passed in, the default value is used which is the current DID itself.
- **priority_requirement** (*int, optional*) – A non-negative integer showing the minimum hierarchical level a key must have in order to remove this key.

export_encrypted_keys_as_json(*password*)
Exports encrypted keys as JSON.

Parameters **password** (*str*) – A password to use for the encryption of the keys.

Returns Encrypted keys JSON.

Return type str

export_encrypted_keys_as_str(*password*)
Exports encrypted keys cipher text.

Parameters **password** (*str*) – A password to use for the encryption of the keys.

Returns Encrypted keys cipher text.

Return type str

export_entry_data()
Exports content that can be recorded on-chain to create the DID.

Returns A dictionary with the ExtIDs and entry content of strings used that are the header columns.

Return type dict

Raises ValueError – If there are no management keys. If there is no management key with priority 0. If the entry size exceeds the entry size limit.

get_chain()

Returns The chain ID where this DID is (or will be) stored

Return type str

id

static is_valid_did(*did*)

mainnet()
Sets the DID network to mainnet.

management_key(*alias*, *priority*, *key_type=<KeyType.EdDSA: 'Ed25519VerificationKey'>*, *controller=None*, *priority_requirement=None*)
Creates a new management key for the DID.

Parameters

- **alias** (*str*) – A human-readable nickname for the key. It should be unique across the keys defined in the DID document.
- **priority** (*int*) – A non-negative integer showing the hierarchical level of the key. Keys with lower priority override keys with higher priority.
- **key_type** (KeyType, *optional*) – Identifies the type of signature that the key pair can be used to generate and verify.
- **controller** (*str, optional*) – An entity that controls the key. It must be a valid DID. If the argument is not passed in, the default value is used which is the current DID itself.
- **priority_requirement** (*int, optional*) – A non-negative integer showing the minimum hierarchical level a key must have in order to remove this key.

method_spec_version_upgrade(*new_spec_version*)

Parameters **new_spec_version** (*str*) – The new DID Method version

Raises

- RuntimeError – If no management keys are available for the DID
- ValueError – If the new version is not an upgrade on the current version

Returns

Return type *DIDVersionUpgrader*

record_on_chain(*factomd*, *walletd*, *ec_address*, *verbose=False*)

Attempts to create the DIDManagement chain.

Parameters

- **factomd** (*obj*) – Factomd instance, instantiated from the Python factom-api package.
- **walletd** (*obj*) – Factom walletd instance, instantiated from the Python factom-api package.
- **ec_address** (*str*) – EC address used to pay for the chain & entry creation.
- **verbose** (*bool, optional*) – If true, display the contents of the entry that will be recorded on-chain.

Raises RuntimeError – If the chain cannot be created

service(*alias*, *service_type*, *endpoint*, *priority_requirement=None*)

Adds a new service to the DID Document.

Parameters

- **alias** (*str*) – A human-readable nickname for the service endpoint. It should be unique across the services defined in the DID document.
- **service_type** (*str*) – Type of the service endpoint.
- **endpoint** (*str*) – A service endpoint may represent any type of service the subject wishes to advertise, including decentralized identity management services for further discovery, authentication, authorization, or interaction. The service endpoint must be a valid URL.
- **priority_requirement** (*int, optional*) – A non-negative integer showing the minimum hierarchical level a key must have in order to remove this service.

testnet()
Sets the DID network to testnet.

update()

Raises RuntimeError – If no management keys are available for the DID

Returns An object allowing updates to the existing DID

Return type *DIDUpdater*

CHAPTER 3

client.encryptor module

factom_did.client.encryptor.**encrypt_keys**(*management_keys*, *did_keys*, *password*)

Encrypts keys with a password.

Parameters

- **management_keys** (*ManagementKeyModel[]*) – A list of management keys to be encrypted.
- **did_keys** (*DidKeyModel[]*) – A list of did keys to be encrypted.
- **password** (*str*) – A password to use for the encryption of the keys.

Returns An object containing salt, initial vector, tag and encrypted data.

Return type obj

factom_did.client.encryptor.**decrypt_keys_from_str**(*cipher_text_b64*, *password*, *encryption_algo='AES-GCM'*)

Decrypts keys from cipher text and password.

Parameters

- **cipher_text_b64** (*str*) – Base 64 encoded cipher text.
- **password** (*str*) – A password used for the encryption of the keys.
- **encryption_algo** (*str*) – The encryption algorithm used. Currently only 'AES-GCM' is supported

Returns An object containing dictionaries of decrypted management and did keys.

Return type obj

Raises ValueError – If the cipher text or the password used for the encryption is invalid.

factom_did.client.encryptor.**decrypt_keys_from_json_str**(*encrypted_keys_json_str*, *password*)

Decrypts keys from JSON string and password. The JSON string must have a schema compatible with the one produced by DID.export_encrypted_keys_as_json():

‘{

“encryptionAlgo”: { “salt”: . . . , “iv”: . . . , “name”: . . . , “tagLength”: . . . ,

}, “data”: . . . (encrypted private keys), “did”: . . .

}’

Parameters

- **encrypted_keys_json_str** (*str*) – JSON string containing encrypted keys data.
- **password** (*str*) – A password used for the encryption of the keys.

Returns An object containing dictionaries of decrypted management and did keys.

Return type obj

Raises ValueError – If the JSON or the password used for the encryption is invalid.

factom_did.client.encryptor.**decrypt_keys_from_json_file**(*file_path*, *password*)

Decrypts keys from JSON file and password. The file must contain valid JSON with a schema compatible with the one produced by DID.export_encrypted_keys_as_json(). See decrypt_keys_from_json_str for details.

Parameters

- **file_path** (*str*) – Path to a file to read from.
- **password** (*str*) – A password used for the encryption of the keys.

Returns An object containing dictionaries of decrypted management and did keys.

Return type obj

Raises ValueError – If the file or the password is invalid.

CHAPTER 4

client.enums module

class factom_did.client.enums.**KeyType**
Bases: enum.Enum

An enumeration.

ECDSA = 'ECDSASecp256k1VerificationKey'

EdDSA = 'Ed25519VerificationKey'

RSA = 'RSAVerificationKey'

from_str = <function KeyType.from_str>

class factom_did.client.enums.**EntryType**
Bases: enum.Enum

An enumeration.

Create = 'DIDManagement'

Deactivation = 'DIDDeactivation'

Update = 'DIDUpdate'

VersionUpgrade = 'DIDMethodVersionUpgrade'

class factom_did.client.enums.**DIDKeyPurpose**
Bases: enum.Enum

An enumeration.

AuthenticationKey = 'authentication'

PublicKey = 'publicKey'

from_str = <function DIDKeyPurpose.from_str>

class factom_did.client.enums.**Network**
Bases: enum.Enum

An enumeration.

```
    Mainnet = 'mainnet'
    Testnet = 'testnet'
    Unspecified = ''
    from_str = <function Network.from_str>
```

CHAPTER 5

client.keys package

5.1 Submodules

CHAPTER 6

client.service module

class factom_did.client.service.**Service**(*alias*, *service_type*, *endpoint*, *priority_requirement=None*)

Bases: object

Represent a service associated with a DID. A service is an end-point, which can be used to communicate with the DID or to carry out different tasks on behalf of the DID (such as signatures, e.g.)

alias
A human-readable nickname for the service endpoint.

Type str

service_type
Type of the service endpoint (e.g. email, credential store).

Type str

endpoint
A service endpoint may represent any type of service the subject wishes to advertise, including decentralized identity management services for further discovery, authentication, authorization, or interaction. The service endpoint must be a valid URL.

Type str

priority_requirement
A non-negative integer showing the minimum hierarchical level a key must have in order to remove this service.

Type int, optional

static from_entry_dict(*entry_dict*, *version='1.0.0'*)

full_id(*did*)

Returns The full id for the service, constituting of the DID_METHOD_NAME, the controller and the service alias.

Return type str

to_entry_dict(*did*, *version='1.0.0'*)

Converts the object to a dictionary suitable for recording on-chain.

Parameters

- **did** (*str*) – The DID to which this service belongs.
- **version** (*str*) – The entry schema version

Raises NotImplementedError – If the entry schema version is not supported

CHAPTER 7

client.updater module

class factom_did.client.updater.**DIDUpdater**(*did*)

Bases: object

Facilitates the creation of an update entry for an existing DID.

Provides support for adding and revoking management keys, DID keys and services.

did

The DID object to update

Type *client.did.DID*

add_did_key(*alias*, *purpose*, *key_type=<KeyType.EdDSA: 'Ed25519VerificationKey'>*, *controller=None*, *priority_requirement=None*)

Adds a DID key to the DID object.

Parameters

- **alias** (*str*) –
- **purpose** (*did.enums.DIDKeyPurpose*) –
- **key_type** (KeyType, *optional*) –
- **controller** (*str, optional*) –
- **priority_requirement** (*int, optional*) –

add_management_key(*alias*, *priority*, *key_type=<KeyType.EdDSA: 'Ed25519VerificationKey'>*, *controller=None*, *priority_requirement=None*)

Adds a management key to the DID object.

Parameters

- **alias** (*str*) –
- **priority** (*int*) –
- **key_type** (KeyType, *optional*) –
- **controller** (*str, optional*) –

- **priority_requirement** (*int, optional*) –

add_service(*alias*, *service_type*, *endpoint*, *priority_requirement=None*)
Adds a service to the DID object.

Parameters

- **alias** (*str*) –
- **service_type** (*str*) –
- **endpoint** (*str*) –
- **priority_requirement** (*int, optional*) –

static exists_management_key_with_priority_zero(*active_management_keys*, *new_management_keys*, *management_keys_to_revoke*)
Checks if a management key of priority zero would be present if the management keys will be updated according to the given parameters.

Parameters

- **active_management_keys** (*set*) – The currently active management keys
- **new_management_keys** (*set*) – The management keys to be added
- **management_keys_to_revoke** (*set*) – The management keys to be revoked

Returns

Return type bool

export_entry_data()
Constructs a signed DIDUpdate entry ready for recording on-chain.

Returns A dictionary with ExtIDs and content for the entry

Return type dict

Raises RuntimeError – If a management key of sufficient priority is not available to sign the update.

get_updated()

record_on_chain(*factomd*, *walletd*, *ec_address*, *verbose=False*)
Attempts to record the DIDUpdate entry on-chain.

Parameters

- **factomd** (*obj*) – Factomd instance, instantiated from the Python factom-api package.
- **walletd** (*obj*) – Factom walletd instance, instantiated from the Python factom-api package.
- **ec_address** (*str*) – EC address used to pay for the chain & entry creation.
- **verbose** (*bool, optional*) – If true, display the contents of the entry that will be recorded on-chain.

Raises RuntimeError – If the entry cannot be recorded

revoke_did_key(*alias*)
Revokes a DID key from the DID object.

Parameters **alias** (*str*) – The alias of the key to be revoked

revoke_did_key_purpose(*alias*, *purpose*)
Revokes a single purpose of a DID key from DID object.

Parameters

- **alias** (*str*) – The alias of the DID key
- **purpose** (DIDKeyPurpose) – The purpose to revoke

revoke_management_key(*alias*)
Revokes a management key from the DID object.

Parameters **alias** (*str*) – The alias of the key to be revoked

revoke_service(*alias*)
Revokes a service from the DID object.

Parameters **alias** (*str*) – The alias of the service to be revoked

rotate_did_key(*alias*)
Rotates a DID key.

Parameters **alias** (*str*) – The alias of the DID key to be rotated

rotate_management_key(*alias*)
Rotates a management key.

Parameters **alias** (*str*) – The alias of the management key to be rotated

Python Module Index

f

Index

F

G

I

K

M

N

O

P

R

S

T

U

V