
fabric-sdk-py

Release 0.9.0

Hyperledger Community

Jul 09, 2020

CONTENTS

1	Fabric-SDK-Py	1
2	Pre-requisites	3
3	Creating a Virtual Environment	5
4	Run Integration Testing	7
5	Reporting Bug	9
6	Ask Questions	11
7	Get started and read our Documentation	13
8	Feel free to contribute	15
9	Some Important Links	17
10	License	19
11	Package Reference	21
	Python Module Index	161
	Index	163

FABRIC-SDK-PY

Fabric-SDK-Py is the Python 3.x implementation of Hyperledger Fabric SDK. Currently, it mainly supports Fabric 1.4.x version.

PRE-REQUISITES

The SDK requires `Python3` and `Libssl` packages. Run the following commands to install the pre-requisites:

CREATING A VIRTUAL ENVIRONMENT

Virtual environment helps providing a clean environment for making changes and testing them locally. Also, it is highly suggested for testing purposes.

```
$ pip3 install virtualenv # install the virtualenv tool if not installed
$ make venv # create a virtual env
$ source venv/bin/activate
$ # Do the testing here
$ deactivate # deactivate the virtual env
$ make clean # clean the temporary files
```


RUN INTEGRATION TESTING

The following command will run the testing.

```
$ make check # Check environment and run tests
$ make test # Only run test cases
$ tox -e py3 -- test/integration/ca_test.py # Run specified test case
```


REPORTING BUG

We welcome any kind of contribution, You can see open issues, requests, code components and also report an issue [here](#).

ASK QUESTIONS

We are an opensource community and always welcome your questions. Feel free to ask any question on our community [here](#).

GET STARTED AND READ OUR DOCUMENTATION

You can find our documentation [here](#).

FEEL FREE TO CONTRIBUTE

Let's get started and contribute to Fabric-SDK-Py! You can start [here](#).

SOME IMPORTANT LINKS

LICENSE

The Hyperledger Fabric-SDK-Py software uses the [Apache License Version 2.0](#) software license. This document is licensed under a [Creative Commons Attribution 4.0 International License](#).

PACKAGE REFERENCE

11.1 Tutorial of Using Fabric Python SDK

TLDR, run a quick testing.

```
$ HLF_VERSION=1.4.6
$ docker pull hyperledger/fabric-peer:${HLF_VERSION} \
  && docker pull hyperledger/fabric-orderer:${HLF_VERSION} \
  && docker pull hyperledger/fabric-ca:${HLF_VERSION} \
  && docker pull hyperledger/fabric-ccenv:${HLF_VERSION}
$ docker-compose -f test/fixtures/docker-compose-2orgs-4peers-tls.yaml up
$ pip3 install virtualenv; make venv
$ source venv/bin/activate
$ make install
$ tox -e py3 -- test/integration/e2e_test.py # Run specified test case
$ deactivate
```

11.1.1 0. Prepare a Testing Environment

0.1. Install Fabric SDK

```
$ git clone https://github.com/hyperledger/fabric-sdk-py.git
$ cd fabric-sdk-py
$ make install
```

Optionally, you can also verify the version number or run all testing cases.

```
$ python
>>> import hfc
>>> print(hfc.VERSION)
0.9.0
>>> exit()

$ make check
```

0.2. Setup a Fabric Network

If you already have a running fabric network, ignore this.

To start an example fabric network you can simply run the following command:

```
$ HLF_VERSION=1.4.6
$ docker pull hyperledger/fabric-peer:${HLF_VERSION}
$ docker pull hyperledger/fabric-orderer:${HLF_VERSION}
$ docker pull hyperledger/fabric-ca:${HLF_VERSION}
$ docker pull hyperledger/fabric-ccenv:${HLF_VERSION}
$ docker-compose -f test/fixtures/docker-compose-2orgs-4peers-tls.yaml up
```

Then you'll have a fabric network with 3 organizations, 4 peers and 1 orderer:

- org1.example.com
 - peer0.org1.example.com
 - peer1.org1.example.com
- org2.example.com
 - peer0.org2.example.com
 - peer1.org2.example.com
- orderer.example.com
 - orderer.example.com
- Note: make sure `configtxgen` is in the '\$PATH'.

If you want to understand more details on starting up a fabric network, feel free to see the [Building Your First Network](#) tutorial.

0.3. Create the Connection Profile

A network connection profile helps SDK connect to the fabric network by providing all required information to operate with a fabric network, including:

- Client credentials file location;
- Service endpoints for peer, orderer and ca;

The `network.json` is an example, please modify the content accordingly.

Now you can use the Python SDK to work with the fabric network!

11.1.2 1. Get Credentials

1.1 Load the Connection Profile

Load all network information from the profile, and check the resources.

```
from hfc.fabric import Client

cli = Client(net_profile="test/fixtures/network.json")

print(cli.organizations) # orgs in the network
```

(continues on next page)

(continued from previous page)

```
print(cli.peers) # peers in the network
print(cli.orderers) # orderers in the network
print(cli.CAs) # ca nodes in the network
```

1.2 Prepare User Id (Optionally)

SDK need the credential file as a valid network user.

Typically there are two ways: using cryptogen or using Fabric-CA. That will depend on how your network boots up with.

1.2.1 Using Local Credential

SDK will load the valid credential from local path (the credential files must be put there in advance).

```
from hfc.fabric import Client

cli = Client(net_profile="test/fixtures/network.json")
org1_admin = cli.get_user(org_name='org1.example.com', name='Admin') # get the admin_
↳user from local path
```

1.2.2 Get Credential from Fabric CA

Here demos how to interact with Fabric CA.

- Enroll into Fabric CA with admin role;
- Register a user user1;
- Enroll with the new user user1 and get local credential;
- Re-enroll the user1;
- Revoke the user1.

To use CA, a CA server must be started. For example,

```
$ docker-compose -f test/fixtures/ca/docker-compose.yml up
```

```
from hfc.fabric_ca.caservice import ca_service

casvc = ca_service(target="http://127.0.0.1:7054")
adminEnrollment = casvc.enroll("admin", "adminpw") # now local will have the admin_
↳enrollment
secret = adminEnrollment.register("user1") # register a user to ca
user1Enrollment = casvc.enroll("user1", secret) # now local will have the user_
↳enrollment
user1ReEnrollment = casvc.reenroll(user1Enrollment) # now local will have the user_
↳reenrolled object
RevokedCerts, CRL = adminEnrollment.revoke("user1") # revoke the user if you need
```

You can also use the new identity management system:

```

from hfc.fabric_ca.caservice import ca_service

casvc = ca_service(target="http://127.0.0.1:7054")
identityService = casvc.newIdentityService()

admin = casvc.enroll("admin", "adminpw") # now local will have the admin user
secret = identityService.create(admin, 'foo') # create user foo
res = identityService.getOne('foo', admin) # get user foo
res = identityService.getAll(admin) # get all users
res = identityService.update('foo', admin, maxEnrollments=3, affiliation='.',
↪ enrollmentSecret='bar') # update user foo
res = identityService.delete('foo', admin) # delete user foo

```

You can also store the newly created credentials in the FileSystemWallet:

```

from hfc.fabric_ca.caservice import ca_service
from hfc.fabric_network import wallet

casvc = ca_service(target="http://127.0.0.1:7054")
adminEnrollment = casvc.enroll("admin", "adminpw") # now local will have the admin_
↪ enrollment
secret = adminEnrollment.register("user1") # register a user to ca
user1Enrollment = casvc.enroll("user1", secret) # now local will have the user_
↪ enrollment
new_wallet = wallet.FileSystemWallet() # Creates default wallet at ./tmp/hfc-kvs
user_identity = wallet.Identity("user1", user1Enrollment) # Creates a new Identity of_
↪ the enrolled user
user_identity.CreateIdentity(new_wallet) # Stores this identity in the_
↪ FileSystemWallet
user1 = new_wallet.create_user("user1", "Org1", "Org1MSP") # Returns an instance of_
↪ the user object with the newly created credentials

```

You can also store the newly created credentials in the InMemoryWallet:

```

from hfc.fabric_ca.caservice import ca_service
from hfc.fabric_network import inmemorywallet

casvc = ca_service(target="http://127.0.0.1:7054")
adminEnrollment = casvc.enroll("admin", "adminpw") # now local will have the admin_
↪ enrollment
secret = adminEnrollment.register("user1") # register a user to ca
user1Enrollment = casvc.enroll("user1", secret) # now local will have the user_
↪ enrollment
new_wallet = inmemorywallet.InMemoryWallet() # Creates a new instance of the class_
↪ InMemoryWallet
new_wallet.put("user1", user1Enrollment) # Saves the credentials of 'user1' in the_
↪ wallet

```

11.1.3 2. Operate Channels with Fabric Network

2.1 Create a new channel and join it

Use sdk to create a new channel and let peers join it.

```
import asyncio
from hfc.fabric import Client

loop = asyncio.get_event_loop()

cli = Client(net_profile="test/fixtures/network.json")
org1_admin = cli.get_user(org_name='org1.example.com', name='Admin')

# Create a New Channel, the response should be true if succeed
response = loop.run_until_complete(cli.channel_create(
    orderer='orderer.example.com',
    channel_name='businesschannel',
    requestor=org1_admin,
    config_yaml='test/fixtures/e2e_cli/',
    channel_profile='TwoOrgsChannel'
))
print(response == True)

# Join Peers into Channel, the response should be true if succeed
orderer_admin = cli.get_user(org_name='orderer.example.com', name='Admin')
responses = loop.run_until_complete(cli.channel_join(
    requestor=org1_admin,
    channel_name='businesschannel',
    peers=['peer0.org1.example.com',
           'peer1.org1.example.com'],
    orderer='orderer.example.com'
))
print(len(responses) == 2)

# Join Peers from a different MSP into Channel
org2_admin = cli.get_user(org_name='org2.example.com', name='Admin')

# For operations on peers from org2.example.com, org2_admin is required as requestor
responses = loop.run_until_complete(cli.channel_join(
    requestor=org2_admin,
    channel_name='businesschannel',
    peers=['peer0.org2.example.com',
           'peer1.org2.example.com'],
    orderer='orderer.example.com'
))
print(len(responses) == 2)
```

2.2 Update the Channel Configuration

```
import asyncio
from hfc.fabric import Client

loop = asyncio.get_event_loop()

cli = Client(net_profile="test/fixtures/network.json")
org1_admin = cli.get_user(org_name='org1.example.com', name='Admin')

config_tx_file = './configtx.yaml'

orderer_admin = cli.get_user(org_name='orderer.example.com', name='Admin')
loop.run_until_complete(cli.channel_update(
    orderer='orderer.example.com',
    channel_name='businesschannel',
    requestor=orderer_admin,
    config_tx=config_tx_file))
```

11.1.4 3. Operate Chaincodes with Fabric Network

Use sdk to install, instantiate and invoke chaincode.

```
import asyncio
from hfc.fabric import Client

loop = asyncio.get_event_loop()

cli = Client(net_profile="test/fixtures/network.json")
org1_admin = cli.get_user('org1.example.com', 'Admin')

# Make the client know there is a channel in the network
cli.new_channel('businesschannel')

# Install Example Chaincode to Peers
# GOPATH setting is only needed to use the example chaincode inside sdk
import os
gopath_bak = os.environ.get('GOPATH', '')
gopath = os.path.normpath(os.path.join(
    os.path.dirname(os.path.realpath('__file__')),
    'test/fixtures/chaincode'
))
os.environ['GOPATH'] = os.path.abspath(gopath)

# The response should be true if succeed
responses = loop.run_until_complete(cli.chaincode_install(
    requestor=org1_admin,
    peers=['peer0.org1.example.com',
           'peer1.org1.example.com'],
    cc_path='github.com/example_cc',
    cc_name='example_cc',
    cc_version='v1.0'
))

# Instantiate Chaincode in Channel, the response should be true if succeed
args = ['a', '200', 'b', '300']
```

(continues on next page)

(continued from previous page)

```

# policy, see https://hyperledger-fabric.readthedocs.io/en/release-1.4/endorsement-
↳policies.html
policy = {
    'identities': [
        {'role': {'name': 'member', 'mspId': 'Org1MSP'}},
    ],
    'policy': {
        '1-of': [
            {'signed-by': 0},
        ]
    }
}

response = loop.run_until_complete(cli.chaincode_instantiate(
    requestor=org1_admin,
    channel_name='businesschannel',
    peers=['peer0.org1.example.com'],
    args=args,
    cc_name='example_cc',
    cc_version='v1.0',
    cc_endorsement_policy=policy, # optional, but recommended
    collections_config=None, # optional, for private data policy
    transient_map=None, # optional, for private data
    wait_for_event=True # optional, for being sure chaincode is_
↳instantiated
))

# Invoke a chaincode
args = ['a', 'b', '100']
# The response should be true if succeed
response = loop.run_until_complete(cli.chaincode_invoke(
    requestor=org1_admin,
    channel_name='businesschannel',
    peers=['peer0.org1.example.com'],
    args=args,
    cc_name='example_cc',
    transient_map=None, # optional, for private data
    wait_for_event=True, # for being sure chaincode invocation has been_
↳committed in the ledger, default is on tx event
    #cc_pattern='^invoked*' # if you want to wait for chaincode event and_
↳you have a `stub.SetEvent("invoked", value)` in your chaincode
))

# Query a chaincode
args = ['b']
# The response should be true if succeed
response = loop.run_until_complete(cli.chaincode_query(
    requestor=org1_admin,
    channel_name='businesschannel',
    peers=['peer0.org1.example.com'],
    args=args,
    cc_name='example_cc'
))

# Upgrade a chaincode
# policy, see https://hyperledger-fabric.readthedocs.io/en/release-1.4/endorsement-
↳policies.html

```

(continues on next page)

(continued from previous page)

```

policy = {
    'identities': [
        {'role': {'name': 'member', 'mspId': 'Org1MSP'}},
        {'role': {'name': 'admin', 'mspId': 'Org1MSP'}},
    ],
    'policy': {
        '1-of': [
            {'signed-by': 0}, {'signed-by': 1},
        ]
    }
}
response = loop.run_until_complete(cli.chaincode_upgrade(
    requestor=org1_admin,
    channel_name='businesschannel',
    peers=['peer0.org1.example.com'],
    args=args,
    cc_name='example_cc',
    cc_version='v1.0',
    cc_endorsement_policy=policy, # optional, but recommended
    collections_config=None, # optional, for private data policy
    transient_map=None, # optional, for private data
    wait_for_event=True # optional, for being sure chaincode is upgraded
))

```

You can also invoke and query the chaincode through the Gateway This has to be done after installing and instantiating the chaincode

```

import asyncio
from hfc.fabric_network.gateway import Gateway
from hfc.fabric_network.network import Network
from hfc.fabric_network.contract import Contract
from hfc.fabric import Client

loop = asyncio.get_event_loop()

cli = Client(net_profile="test/fixtures/network.json")
org1_admin = cli.get_user(org_name='org1.example.com', name='Admin')

new_gateway = Gateway() # Creates a new gateway instance
options = {'wallet': ''}
response = loop.run_until_complete(new_gateway.connect('test/fixtures/network.json',
↳options))
new_network = loop.run_until_complete(new_gateway.get_network('businesschannel', org1_
↳admin))
new_contract = new_network.get_contract('example_cc')
response = loop.run_until_complete(new_contract.submit_transaction('businesschannel',
↳['a', 'b', '100'], org1_admin))
response = loop.run_until_complete(new_contract.evaluate_transaction('businesschannel
↳', ['b'], org1_admin))

```


11.1.5 4. Query Informations

By default, `query` methods returns a decoded response.

If you need to get the raw response from the ledger you can add `decode=False` param.

4.1 Basic Usage

```
import asyncio
from hfc.fabric import Client

loop = asyncio.get_event_loop()
cli = Client(net_profile="test/fixtures/network.json")
org1_admin = cli.get_user('org1.example.com', 'Admin')

# Query Peer installed chaincodes, make sure the chaincode is installed
response = loop.run_until_complete(cli.query_installed_chaincodes(
    requestor=org1_admin,
    peers=['peer0.org1.example.com'],
    decode=True
))

"""
# An example response:

chaincodes {
  name: "example_cc"
  version: "v1.0"
  path: "github.com/example_cc"
  id: "\374\361\027j(\332\225\367\253\030\242\303U&\356\326\241\2003/\033\266:\314\
↪250\032\254\221L#\006G"
}
"""

# Query Peer Joined channel
response = loop.run_until_complete(cli.query_channels(
    requestor=org1_admin,
    peers=['peer0.org1.example.com'],
    decode=True
))

"""
# An example response:

channels {
  channel_id: "businesschannel"
}
"""
```

4.2 Query Block by block hash & transaction id

```

import asyncio
from hfc.fabric import Client

loop = asyncio.get_event_loop()
cli = Client(net_profile="test/fixtures/network.json")
org1_admin = cli.get_user('org1.example.com', 'Admin')

# first get the hash by calling 'query_info'
response = loop.run_until_complete(cli.query_info(
    requestor=org1_admin,
    channel_name='businesschannel',
    peers=['peer0.org1.example.com'],
    decode=True
))

"""
# An example response:

height: 3
currentBlockHash: "\\|255\\317\\341$\\|371\\242aP\\030u\\325~\\263!\\352G\\014\\007\\353\\353\\
↪247\\235<\\353\\020\\026\\345\\254\\252r"
previousBlockHash: "\\324\\214\\275z\\301)\\351\\224 \\225\\306\\"\\250jBMa\\3432r\\035\\023\\310\\
↪250\\017w\\013\\303!f\\340\\272"
"""

test_hash = response.currentBlockHash

response = loop.run_until_complete(cli.query_block_by_hash(
    requestor=org1_admin,
    channel_name='businesschannel',
    peers=['peer0.org1.example.com'],
    block_hash=test_hash,
    decode=True
))

tx_id = response.get('data').get('data')[0].get(
    'payload').get('header').get(
    'channel_header').get('tx_id')

response = loop.run_until_complete(cli.query_block_by_txid(
    requestor=org1_admin,
    channel_name='businesschannel',
    peers=['peer0.org1.example.com'],
    tx_id=tx_id,
    decode=True
))

```

4.3 Query Block, Transaction and Instantiated Chaincodes

```

import asyncio
from hfc.fabric import Client

loop = asyncio.get_event_loop()
cli = Client(net_profile="test/fixtures/network.json")
org1_admin = cli.get_user('org1.example.com', 'Admin')

# Query Block by block number
response = loop.run_until_complete(cli.query_block(
    requestor=org1_admin,
    channel_name='businesschannel',
    peers=['peer0.org1.example.com'],
    block_number='1',
    decode=True
))

# Query Transaction by tx id
# example txid of instantiated chaincode transaction
response = loop.run_until_complete(cli.query_transaction(
    requestor=org1_admin,
    channel_name='businesschannel',
    peers=['peer0.org1.example.com'],
    tx_id=tx_id, # tx_id same at 4.2
    decode=True
))

# Query Instantiated Chaincodes
response = loop.run_until_complete(cli.query_instantiated_chaincodes(
    requestor=org1_admin,
    channel_name='businesschannel',
    peers=['peer0.org1.example.com'],
    decode=True
))

```

4.4 Get channel configuration

```

import asyncio
from hfc.fabric import Client

loop = asyncio.get_event_loop()
cli = Client(net_profile="test/fixtures/network.json")
org1_admin = cli.get_user('org1.example.com', 'Admin')

# Get channel config
response = loop.run_until_complete(cli.get_channel_config(
    requestor=org1_admin,
    channel_name='businesschannel',
    peers=['peer0.org1.example.com'],
    decode=True
))

```

4.5 Use channel discovery

```
import asyncio
from hfc.fabric import Client

loop = asyncio.get_event_loop()
cli = Client(net_profile="test/fixtures/network.json")
org1_admin = cli.get_user('org1.example.com', 'Admin')

# Get config from local channel discovery
response = loop.run_until_complete(cli.query_peers(
    requestor=org1_admin,
    peer='peer0.org1.example.com',
    channel='businesschannel',
    local=True,
    decode=True
))

# Get config from channel discovery over the network
response = loop.run_until_complete(cli.query_peers(
    requestor=org1_admin,
    peer='peer0.org1.example.com',
    channel='businesschannel',
    local=False,
    decode=True
))
```

11.1.6 License

This document is licensed under a Creative Commons Attribution 4.0 International License.

11.2 Release Notes

11.2.1 v0.9.0 Feb 1, 2020

Add new features

- Support fabric 1.4.0 network;
- Add channel event hub options;

Improvements

- rework create_seek_info;

Known Vulnerabilities

none

Resolved Vulnerabilities

none

Change Log

<https://github.com/hyperledger/fabric-sdk-py/blob/master/CHANGELOG.md#v0.9.0>

11.2.2 v0.8.1 July 21, 2019

Add new features

- Support fabric 1.4.0 network;
- Add fabric-ca functions;
- Add tls support;

Improvements

- Support async (python 3.6)
- Update channel event hub;
- Update Tutorial;
- Add test cases;

Known Vulnerabilities

none

Resolved Vulnerabilities

none

Change Log

<https://github.com/hyperledger/fabric-sdk-py/blob/master/CHANGELOG.md#v0.8.1>

11.2.3 v0.8.0 Apr 12, 2019

Add new features

- Support fabric 1.x network;
- Add more new features:
 1. Implement event service
 2. Implement discovery service
 3. Replace Rx module with python native async/await
 4. CA supporting.
 5. Implement mutual encryption.

Improvements

- Improve the documentation;
- Add example code;
- Refine invoke api;
- Improve query;
- Tutorials Updated;
- Test Updated;

Known Vulnerabilities

- Private collection support

Resolved Vulnerabilities

none

Change Log

<https://github.com/hyperledger/fabric-sdk-py/blob/master/CHANGELOG.md#v0.8.0>

11.3 Python Coding Style

In order to make the code more maintainable, and helps developers understand code in reviews, we have a set of style guidelines for clarity.

- Please read [pep8 style guide](#) before you try to contribute any code. The guidelines gives some basic coding conventions, including:
 - Code lay-out
 - String Quotes
 - Naming Conventions

- Comments
- Some general guidelines you should follow like following when you write SDK code:
 - Use only UNIX style newlines (`\n`), not Windows style (`\r\n`)
 - It is preferred to wrap long lines in parentheses and not a backslash for line continuation.
 - Do not import more than one module per line
 - Docstrings should not start with a space.
 - Multi line docstrings should end on a new line.
 - Multi line docstrings should start without a leading new line.
 - Multi line docstrings should start with a one line summary followed by an empty line.
- For every api which will be used by our client, the api Docstrings are mandatory. We are here following the Python docstring format. For example:

```
def square_root(n):  
    """Calculate the square root of a number  
  
    Args:  
        n (int): the number to get the square root of  
  
    Returns:  
        square_root (float): the square root of n  
  
    Raises:  
        TypeError: if n is not a number  
        ValueError: if n is negative  
    """  
    pass
```

11.3.1 License

This document is licensed under a Creative Commons Attribution 4.0 International License.

../../CONTRIBUTING.md

11.4 API Reference

This page contains auto-generated API reference documentation¹.

¹ Created with sphinx-autoapi

11.4.1 hfc

Subpackages

`hfc.fabric`

Subpackages

`hfc.fabric.channel`

Submodules

`hfc.fabric.channel.channel`

Module Contents

Classes

<code>Channel(name, client)</code>	The class represents the channel.
------------------------------------	-----------------------------------

Functions

<code>create_system_channel(client, name=SYSTEM_CHANNEL_NAME)</code>	Create system channel instance
<code>create_app_channel(client, name)</code>	Create application channel instance

`hfc.fabric.channel.channel.SYSTEM_CHANNEL_NAME = testchainid`

`hfc.fabric.channel.channel._logger`

class `hfc.fabric.channel.channel.Channel` (*name, client*)

Bases: `object`

The class represents the channel.

This is a client-side-only call. To create a new channel in the fabric call `client._create_or_update_channel()`.

add_orderer (*self, orderer*)

Add orderer endpoint to a channel object.

A channel instance may choose to use a single orderer node, which will broadcast requests to the rest of the orderer network. Or if the application does not trust the orderer nodes, it can choose to use more than one by adding them to the channel instance. And all APIs concerning the orderer will broadcast to all `_orderers` simultaneously.

Parameters `orderer` – an instance of the `Orderer` class

Returns

remove_orderer (*self, orderer*)

Remove orderer endpoint from a channel object.

Parameters `orderer` – an instance of the `Orderer` class

Returns**add_peer** (*self*, *peer*)

Add peer endpoint to a chain object.

Parameters **peer** – an instance of the Peer class**Returns****remove_peer** (*self*, *peer*)

Remove peer endpoint from a channel object.

Parameters **peer** – an instance of the Peer class**Returns****property orderers** (*self*)

Get _orderers of a channel.

Returns The orderer list on the channel**property peers** (*self*)

Get peers of a channel.

Returns The peer list on the chain**property is_dev_mode** (*self*)

Get is_dev_mode

Returns is_dev_mode**_get_latest_block** (*self*, *tx_context*, *orderer*)

Get latest block from orderer.

Parameters

- **tx_context** – a tx_context instance
- **orderer** – a orderer instance

Returns**_get_random_orderer** (*self*)**property name** (*self*)

Get channel name.

Returns channel name**state_store** (*self*)

Get the key val store instance of the instantiating client. Get the KeyValueStore implementation (if any) that is currently associated with this channel :return: the current KeyValueStore associated with this channel / client.

_validate_state (*self*)

Validate channel state.

Raises ValueError –**property is_sys_chan** (*self*)

Get if system channel

_validate_peer (*self*, *peer*)

Validate peer

Parameters **peer** – peer

Raises ValueError –

_validate_peers (*self*, *peers*)
Validate peer set

Parameters peers – peers

Raises ValueError –

send_install_proposal (*self*, *tx_context*, *peers=None*)
Send install chaincode proposal

Parameters

- **install_proposal_req** – install proposal request
- **targets** – a set of peer to send
- **tx_context** – a tx_context instance
- **peers** – peers (Default value = None)

Returns a set of proposal response

_build_channel_header (*type*, *tx_id*, *channel_id*, *timestamp*, *epoch=0*, *extension=None*)
Build channel.

Parameters

- **extension** – extension (Default value = None)
- **timestamp** – timestamp
- **channel_id** – channel id
- **tx_id** – transaction id
- **type** – type
- **epoch** – epoch

Returns common_proto.Header instance (Default value = 0)

is_readonly (*self*)
Check the channel if read-only

Get the channel status to see if the underlying channel has been terminated, making it a read-only channel, where information (transactions and state_store) can be queried but no new transactions can be submitted.

Returns True if the channel is read-only, False otherwise.

join_channel (*self*, *request*)
To join the peer to a channel.

Parameters request – request

Returns A coroutine to handle thanks to asyncio with await asyncio.gather(*responses)

send_instantiate_proposal (*self*, *tx_context*, *peers*)
Send instantiate chaincode proposal.

Parameters

- **tx_context** – transaction context
- **peers** – peers to send this proposal

Returns True in success False in failure

send_upgrade_proposal (*self, tx_context, peers*)

Upgrade the chaincode.

Parameters

- **tx_context** – transaction context
- **peers** – peers to send this proposal

Returns True in success and False in failure

_build_principal (*self, identity*)

_get_policy (*self, policy*)

_check_policy (*self, policy*)

_build_policy (*self, policy, msp=None, returnProto=False*)

_send_cc_proposal (*self, tx_context, command, peers*)

send_tx_proposal (*self, tx_context, peers*)

Invoke the chaincode

Send a transaction proposal to one or more endorser without creating a channel. :param tx_context: transaction context :param peers: the peers to send this proposal

if it is None the channel peers list will be used.

channel_id: channel id client: client context :return: True in success or False in failure.

static _send_tx_proposal (*channel_id, tx_context, peers*)

query_instantiated_chaincodes (*self, tx_context, peers, transient_map=None*)

Parameters

- **tx_context** – tx_context instance
- **peers** – peers in the channel
- **transient_map** – transient map

Returns chain code response (Default value = None)

query_transaction (*self, tx_context, peers, tx_id, transient_map=None*)

Queries the ledger for Transaction by transaction ID.

Parameters

- **tx_context** – tx_context instance
- **peers** – peers in the channel
- **tx_id** – transaction ID (string)
- **transient_map** – transient map

Returns chain code response (Default value = None)

get_block_between (*self, tx_context, orderer, start, end*)

Parameters

- **tx_context** – tx_context instance
- **orderer** – orderer instance
- **start** – id of block to start query for

- **end** – id of block to end query for

Returns block(s)

query_block (*self, tx_context, peers, block_number, transient_map=None*)

Queries the ledger for Block by block number.

Parameters

- **tx_context** – tx_context instance
- **peers** – peers in the channel
- **block_number** – block to query for
- **transient_map** – transient map (Default value = None)

Returns class BlockDecoder

query_block_by_hash (*self, tx_context, peers, block_hash, transient_map=None*)

Parameters

- **tx_context** – tx_context instance
- **peers** – peers in the channel
- **block_hash** – block to query for
- **transient_map** – transient map (Default value = None)

Returns class ChaincodeQueryResponse

query_block_by_txid (*self, tx_context, peers, tx_id, transient_map=None*)

Parameters

- **tx_context** – tx_context instance
- **peers** – peers in the channel
- **tx_id** – transaction id
- **transient_map** – transient map (Default value = None)

Returns class ChaincodeQueryResponse

query_info (*self, tx_context, peers, transient_map=None*)

Query the information of channel

Queries for various useful information on the state of the channel (height, known peers).

Parameters

- **tx_context** – tx_context instance
- **peers** – peers in the channel
- **transient_map** – (Default value = None)

Returns class ChaincodeQueryResponse channelinfo with height, currently the only useful information.

get_channel_config (*self, tx_context, peers, transient_map=None*)

Query the current config block for this channel

Parameters

- **tx_context** – tx_context instance

- **peers** – peers in the channel
- **transient_map** – (Default value = None)

Returns class ChaincodeQueryResponse channelinfo with height, currently the only useful information.

async get_channel_config_with_orderer (*self*, *tx_context*, *orderer*)

Query the current config block for this channel

Parameters

- **tx_context** – tx_context instance
- **peers** – peers in the channel

:return:class ChaincodeQueryResponse channelinfo with height, currently the only useful information.

_discovery (*self*, *requestor*, *target*, *local=False*, *config=False*, *interests=None*)

Send a request from a target peer to discover information about the network

Parameters

- **requestor** (*instance*) – a user to make the request
- **target** (*instance*) – target peer to send discovery request
- **local** (*bool*) – include local endpoints in the query (Default value = False)
- **config** (*bool*) – include channel configuration in the query (Default value = False)
- **interests** (*list*) – interests about an endorsement for cc (Default value = None)

Returns Response from Discovery Service

_build_proto_cc_interest (*self*, *interest*)

Use a list of DiscoveryChaincodeCall to build an interest.

Parameters **interest** –

Returns

newChannelEventHub (*self*, *peer*, *requestor*)

getChannelEventHubsForOrg (*self*, *requestor*, *mspid=None*)

`hfc.fabric.channel.channel.create_system_channel` (*client*,
name=SYSTEM_CHANNEL_NAME)

Create system channel instance

Parameters

- **client** – client instance
- **name** – system channel name (Default value = SYSTEM_CHANNEL_NAME)

Returns return system channel instance

`hfc.fabric.channel.channel.create_app_channel` (*client*, *name*)

Create application channel instance

Parameters

- **client** – client instance
- **name** – return application channel instance

Returns system channel instance

`hfc.fabric.channel.channel_configuration`

Module Contents

Classes

<code>ChannelConfiguration</code> (<i>config=None, file_path=None</i>)	A class represents channel configuration bytes.
--	---

class `hfc.fabric.channel.channel_configuration.ChannelConfiguration` (*config=None, file_path=None*)

Bases: object

A class represents channel configuration bytes.

property `config` (*self*)

Get config bytes. :return: raw config bytes

`hfc.fabric.channel.channel_eventhub`

Module Contents

Classes

<code>EventRegistration</code> (<i>onEvent=None, unregister=True, disconnect=False</i>)	A class represents event registration.
---	--

<code>ChaincodeRegistration</code> (<i>ccid, pattern, er, as_array</i>)	A class represents chaincode registration.
---	--

<code>ChannelEventHub</code> (<i>peer, channel_name, requestor</i>)	A class represents channel event hub.
---	---------------------------------------

`hfc.fabric.channel.channel_eventhub._logger`

`hfc.fabric.channel.channel_eventhub.NO_START_STOP = 0`

`hfc.fabric.channel.channel_eventhub.START_ONLY = 1`

`hfc.fabric.channel.channel_eventhub.END_ONLY = 2`

`hfc.fabric.channel.channel_eventhub.START_AND_END = 3`

class `hfc.fabric.channel.channel_eventhub.EventRegistration` (*onEvent=None, unregister=True, disconnect=False*)

Bases: object

A class represents event registration.

class `hfc.fabric.channel.channel_eventhub.ChaincodeRegistration` (*ccid, pattern, er, as_array*)

Bases: object

A class represents chaincode registration.

class `hfc.fabric.channel.channel_eventhub.ChannelEventHub` (*peer, channel_name, requestor*)

Bases: object

A class represents channel event hub.

property connected (*self*)

Get the connected :return: The connected

_create_seek_info (*self*, *start=None*, *stop=None*)

_get_stream (*self*)

get the events of the channel.

Returns the events in success or None in fail.

check_start_stop_connect (*self*, *start=None*, *stop=None*)

check_start_stop_listener (*self*, *start=None*, *stop=None*)

_processBlockEvents (*self*, *block*)

registerBlockEvent (*self*, *unregister=True*, *start=None*, *stop=None*, *disconnect=False*, *on-Event=None*)

unregisterBlockEvent (*self*, *reg_num*)

handle_filtered_tx (*self*, *block*, *tx_id*, *er*)

handle_full_tx (*self*, *block*, *tx_id*, *er*)

_processTxEvents (*self*, *block*)

registerTxEvent (*self*, *tx_id*, *unregister=None*, *start=None*, *stop=None*, *disconnect=False*, *on-Event=None*)

unregisterTxEvent (*self*, *tx_id*)

_queue_chaincode_event (*self*, *chaincode_event*, *block_number*, *tx_id*, *tx_status*, *all_events*)

handle_filtered_chaincode (*self*, *block*, *all_events*)

_handle_full_chaincode (*self*, *tx*, *block_number*, *tx_id*, *tx_status*, *all_events*)

handle_full_chaincode (*self*, *block*, *all_events*)

_processChaincodeEvents (*self*, *block*)

registerChaincodeEvent (*self*, *ccid*, *pattern*, *unregister=False*, *start=None*, *stop=None*, *as_array=None*, *disconnect=False*, *onEvent=None*)

unregisterChaincodeEvent (*self*, *cr*)

have_registrations (*self*)

_on_end_actions (*self*, *event_reg*, *unregister_action*, *startstop_mode*, *unregister*, *disconnect*)

check_replay_end (*self*)

async handle_stream (*self*, *stream*)

connect (*self*, *filtered=True*, *start=None*, *stop=None*, *as_array=False*, *target=None*, *signed_event=None*)

disconnect (*self*)

hfc.fabric.channel.instantiation

Module Contents

Classes

<code>Instantiation()</code>	Chaincode instantiate transaction proposal handler.
------------------------------	---

Functions

<code>_create_instantiation_proposal(chain, tran_prop_req)</code>	Create a chaincode instantiation proposal
---	---

<code>_instantiate_chaincode(chain, cc_instantiation_request, scheduler=None)</code>	Instantiate chaincode.
--	------------------------

<code>create_instantiation_proposal_req(chaincode_id, chaincode_path, chaincode_version, creator, fcn='init', args=None, nonce=crypto.generate_nonce(24), targets=None)</code>	Create instantiation proposal request.
--	--

<code>chaincode_instantiation(chain)</code>	Create instantiate.
---	---------------------

hfc.fabric.channel.instantiation._logger

class hfc.fabric.channel.instantiation.**Instantiation**

Bases: hfc.fabric.channel.transactionproposals.TransactionProposalHandler

Chaincode instantiate transaction proposal handler.

handle (*self*, *tran_prop_req*, *scheduler=None*)

Execute chaincode instantiation transaction proposal request.

Parameters

- **tran_prop_req** – chaincode instantiation transaction proposal request
- **scheduler** – see rx.Scheduler, defaults to None

Returns An rx.Observer wrapper of chaincode instantiation response

hfc.fabric.channel.instantiation._create_instantiation_proposal (*tran_prop_req*, *chain*)

Create a chaincode instantiation proposal This involves assembling the proposal with the data (chaincodeID, chaincode invocation spec, etc.) and signing it using the private key corresponding to the ECert to sign.

Parameters

- **tran_prop_req** – see TransactionProposalRequest
- **chain** – chain instance

Returns (Proposal): The created Proposal instance or None.

hfc.fabric.channel.instantiation._instantiate_chaincode (*chain*, *cc_instantiation_request*, *scheduler=None*)

Instantiate chaincode.

Parameters

- **chain** – chain instance
- **cc_instantiation_request** – see TransactionProposalRequest
- **scheduler** – see rx.Scheduler, defaults to None

Returns An rx.Observable of instantiation response

```
hfc.fabric.channel.instantiation.create_instantiation_proposal_req(chaincode_id,
                                                                    chain-
                                                                    code_path,
                                                                    chain-
                                                                    code_version,
                                                                    creator,
                                                                    fcn='init',
                                                                    args=None,
                                                                    nonce=crypto.generate_nonce(24),
                                                                    tar-
                                                                    gets=None)
```

Create instantiation proposal request.

Parameters

- **chaincode_id** – chaincode_id
- **chaincode_path** (*str*) – chaincode_path
- **chaincode_version** (*str*) – chaincode_version
- **creator** – user
- **fcn** – chaincode init function, defaults to 'init'
- **args** – init function args, defaults to None
- **nonce** – nonce, defaults to crypto.generate_nonce(24)
- **targets** – peers, defaults to None

Returns see TransactionProposalRequest

```
hfc.fabric.channel.instantiation.chaincode_instantiation(chain)
Create instantiate.
```

Parameters **chain** – chain instance

Returns Instantiate instance

hfc.fabric.channel.invocation

Module Contents

Classes

Invocation()

Chaincode invocation transaction proposal handler.

Functions

<code>_create_invocation_proposal(chain, tran_prop_req)</code>	Create a chaincode invocation proposal
<code>_invoke_chaincode(chain, cc_invocation_request, scheduler=None)</code>	Invoke chaincode.
<code>create_invocation_proposal_req(chaincode_id, chaincode_version, creator, fcn='invoke', args=None, nonce=crypto.generate_nonce(24), targets=None)</code>	Create invocation proposal request.
<code>chaincode_invocation(chain)</code>	Create invocation.

`hfc.fabric.channel.invocation._logger`

class `hfc.fabric.channel.invocation.Invocation`

Bases: `hfc.fabric.channel.transactionproposals.TransactionProposalHandler`

Chaincode invocation transaction proposal handler.

handle (*self*, *tran_prop_req*, *scheduler=None*)

Execute chaincode invocation transaction proposal request.

Parameters

- **scheduler** – see `rx.Scheduler` (Default value = `None`)
- **tran_prop_req** – chaincode invocation transaction proposal request

Returns An `rx.Observer` wrapper of chaincode invocation response

`hfc.fabric.channel.invocation._create_invocation_proposal(tran_prop_req, chain)`

Create a chaincode invocation proposal

This involves assembling the proposal with the data (chaincodeID, chaincode invocation spec, etc.) and signing it using the private key corresponding to the ECert to sign.

Parameters

- **tran_prop_req** – see `TransactionProposalRequest`
- **chain** – chain instance

Returns The created Proposal instance or `None`.

`hfc.fabric.channel.invocation._invoke_chaincode(chain, cc_invocation_request, scheduler=None)`

Invoke chaincode.

Parameters

- **chain** – chain instance
- **scheduler** – see `rx.Scheduler` (Default value = `None`)
- **cc_invocation_request** – see `TransactionProposalRequest`

Returns An `rx.Observable` of invocation response

```
hfc.fabric.channel.invocation.create_invocation_proposal_req(chaincode_id,
                                                            chaincode_version,
                                                            creator,
                                                            fcn='invoke',
                                                            args=None,
                                                            nonce=crypto.generate_nonce(24),
                                                            targets=None)
```

Create invocation proposal request.

Parameters

- **fcn** – chaincode invoke function (Default value = ‘invoke’)
- **args** – invoke function args (Default value = None)
- **targets** – peers (Default value = None)
- **nonce** – nonce (Default value = crypto.generate_nonce(24))
- **chaincode_id** – chaincode_id
- **chaincode_version** – chaincode_version
- **creator** – user

Returns see TransactionProposalRequest

```
hfc.fabric.channel.invocation.chaincode_invocation(chain)
```

Create invocation.

Parameters **chain** – chain instance

Returns Invocation instance

Package Contents

Classes

<i>NullHandler</i> (level=NOTSET)	Handler instances dispatch logging events to specific destinations.
-----------------------------------	---

class hfc.fabric.channel.**NullHandler** (*level=NOTSET*)

Bases: logging.Handler

Handler instances dispatch logging events to specific destinations.

The base handler class. Acts as a placeholder which defines the Handler interface. Handlers can optionally use Formatter instances to format records as desired. By default, no formatter is specified; in this case, the ‘raw’ message as determined by record.message is logged.

emit (*self, record*)

Do whatever it takes to actually log the specified logging record.

This version is intended to be implemented by subclasses and so raises a NotImplementedError.

`hfc.fabric.config`

Submodules

`hfc.fabric.config.default`

Module Contents

`hfc.fabric.config.default.DEFAULT`

`hfc.fabric.msp`

`hfc.fabric.transaction`

Submodules

`hfc.fabric.transaction.tx_context`

Module Contents

Classes

`TXContext`(*user*, *crypto*, *tx_prop_req*, *prop_wait_time=-1*) A class represent Transaction context.

Functions

<code>validate</code> (<i>tx_context</i>)	Validate transaction context
<code>create_tx_context</code> (<i>user</i> , <i>crypto</i> , <i>tx_prop_req</i> , <i>prop_wait_time=-1</i>)	Create transaction context

class `hfc.fabric.transaction.tx_context.TXContext` (*user*, *crypto*, *tx_prop_req*, *prop_wait_time=-1*)

Bases: object

A class represent Transaction context.

get_attrs (*self*)

__str__ (*self*)
Return str(self).

property tx_id (*self*)
Get transaction id.

property epoch (*self*)
Get epoch.

property nonce (*self*)
Get nonce

property identity (*self*)

Get identity

sign (*self*, *plain_text*)

Sign the text

Parameters *plain_text* – plain text

Returns the signed text

property prop_wait_time (*self*)

Get proposal wait time

property tx_prop_req (*self*)

Get transaction proposal request

property user (*self*)

Get request user

property crypto (*self*)

Get

`hfc.fabric.transaction.tx_context.validate` (*tx_context*)

Validate transaction context

Parameters *tx_context* – transaction context

Returns transaction context if no error

Raises **ValueError** – Invalid transaction context

`hfc.fabric.transaction.tx_context.create_tx_context` (*user*, *crypto*, *tx_prop_req*,
prop_wait_time=-1)

Create transaction context

Parameters

- **tx_prop_req** – transaction proposal request
- **user** – user
- **crypto** – crypto
- **prop_wait_time** – proposal wait time

Returns a transaction context instance (Default value = -1)

`hfc.fabric.transaction.tx_proposal_request`

Module Contents

Classes

<code>TXProposalRequest</code> (<i>prop_type=None</i> , <i>cc_path=None</i> , <i>cc_type=CC_TYPE_GOLANG</i> , <i>cc_name=None</i> , <i>cc_version=None</i> , <i>fcn=None</i> , <i>args=None</i> , <i>cc_endorsement_policy=None</i> , <i>transient_map=None</i> , <i>packaged_cc=None</i> , <i>collections_config=None</i>)	Class represents transaction proposal request.
---	--

Functions

<code>validate(tx_prop_req)</code>	Check transaction proposal request.
<code>create_tx_prop_req(prop_type=None, cc_path=None, cc_type=CC_TYPE_GOLANG, cc_name=None, cc_version=None, fcn=None, args=None, cc_endorsement_policy=None, transient_map=None, packaged_cc=None, collections_config=None)</code>	Create a transaction proposal request

```
hfc.fabric.transaction.tx_proposal_request.CC_INSTALL = install
```

```
hfc.fabric.transaction.tx_proposal_request.CC_INSTANTIATE = deploy
```

```
hfc.fabric.transaction.tx_proposal_request.CC_INVOKE = invoke
```

```
hfc.fabric.transaction.tx_proposal_request.CC_UPGRADE = upgrade
```

```
hfc.fabric.transaction.tx_proposal_request.CC_QUERY = query
```

```
hfc.fabric.transaction.tx_proposal_request.CC_TYPE_GOLANG = GOLANG
```

```
hfc.fabric.transaction.tx_proposal_request.CC_TYPE_JAVA = JAVA
```

```
hfc.fabric.transaction.tx_proposal_request.CC_TYPE_NODE = NODE
```

```
hfc.fabric.transaction.tx_proposal_request.CC_TYPE_CAR = CAR
```

```
class hfc.fabric.transaction.tx_proposal_request.TXProposalRequest (prop_type=None, cc_path=None, cc_type=CC_TYPE_GOLANG, cc_name=None, cc_version=None, fcn=None, args=None, cc_endorsement_policy=None, transient_map=None, packaged_cc=None, collections_config=None)
```

Bases: object

Class represents transaction proposal request.

```
property cc_type (self)
```

Get chaincode type

Returns return chaincode type

```
property prop_type (self)
```

Get proposal type

Returns return proposal type

```
property cc_path (self)
```

Get chaincode path

Returns return chaincode path

property cc_name (*self*)

Get chaincode name

Returns return chaincode name

property cc_version (*self*)

Get chaincode version

Returns return chaincode version

property fcn (*self*)

Get function name

Returns return function name

property args (*self*)

Get function arguments

Returns return function arguments

property packaged_cc (*self*)

Get packaged chaincode

Returns return packaged chaincode

property cc_endorsement_policy (*self*)

Get endorsement policy

Returns return endorsement policy

property transient_map (*self*)

Get transient map

Returns return transient map

property collections_config (*self*)

Get collections config

Returns return collections config

`hfc.fabric.transaction.tx_proposal_request.validate(tx_prop_req)`

Check transaction proposal request.

Parameters `tx_prop_req` – see TXProposalRequest

Returns transaction proposal request if no error

Raises `ValueError` – Invalid transaction proposal request

`hfc.fabric.transaction.tx_proposal_request.create_tx_prop_req(prop_type=None, cc_path=None, cc_type=CC_TYPE_GOLANG, cc_name=None, cc_version=None, fcn=None, args=None, cc_endorsement_policy=None, transient_map=None, packaged_cc=None, collections_config=None)`

Create a transaction proposal request

Parameters

- **prop_type** – proposal request type (Default value = None)
- **cc_path** – chaincode path (Default value = None)
- **cc_name** – chaincode name (Default value = None)
- **cc_version** – chaincode version (Default value = None)
- **fcn** – function name (Default value = None)
- **args** – function arguments (Default value = None)
- **cc_endorsement_policy** – chaincode endorsement policy (Default value = None)
- **transient_map** – transient data map (Default value = None)
- **packaged_cc** – packaged chaincode source
- **cc_type** – (Default value = CC_TYPE_GOLANG)
- **collections_config** – (Default value = None)

Returns a transaction proposal request (Default value = None)

Package Contents

Classes

<code>NullHandler(level=NOTSET)</code>	Handler instances dispatch logging events to specific destinations.
--	---

class `hfc.fabric.transaction.NullHandler` (*level=NOTSET*)

Bases: `logging.Handler`

Handler instances dispatch logging events to specific destinations.

The base handler class. Acts as a placeholder which defines the Handler interface. Handlers can optionally use Formatter instances to format records as desired. By default, no formatter is specified; in this case, the ‘raw’ message as determined by `record.message` is logged.

emit (*self, record*)

Do whatever it takes to actually log the specified logging record.

This version is intended to be implemented by subclasses and so raises a `NotImplementedError`.

Submodules

`hfc.fabric.block_decoder`

Module Contents

Classes

<i>BlockDecoder()</i>	An object of a fully decoded protobuf message “Block”
<i>FilteredBlockDecoder()</i>	An object of a fully decoded protobuf message “Filtered-Block”
<i>HeaderType()</i>	HeaderType class having decodePayload and convertToString methods

Functions

<i>decode_block_header(proto_block_header)</i>	Decodes the header of Block
<i>decode_block_data(proto_block_data, not_proto=False)</i>	Decodes the data of Block.
<i>decode_block_metadata(proto_block_metadata)</i>	Decodes block metadata from block
<i>decode_block_data_envelope(proto_envelope)</i>	Decodes the envelope contents of Block
<i>decode_header(proto_header)</i>	Decodes the Payload header in envelope
<i>decode_channel_header(header_bytes)</i>	Decodes channel header for Payload channel header
<i>timestamp_to_date(timestamp)</i>	Converts timestamp to current date
<i>decode_version(version_long)</i>	Takes version proto object and returns version
<i>decode_signature_header(signature_header_bytes)</i>	Decode signature header
<i>decode_identity(id_bytes)</i>	Decodes identity
<i>decode_metadata_signatures(metadata_bytes)</i>	Decodes metadata signature from bytes
<i>decode_metadata_value_signatures(proto_metadata_bytes)</i>	Decodes all signatures in metadata values
<i>decode_last_config_sequence_number(metadata_bytes)</i>	Decodes last configuration and index for sequence number
<i>decode_transaction_filter(metadata_bytes)</i>	Decodes transaction filter from metadata bytes
<i>decode_endorser_transaction(trans_bytes)</i>	Decodes
<i>decode_config_envelope(config_envelope_bytes)</i>	Decodes configuration envelope
<i>decode_config(proto_config)</i>	Decodes configuration from config envelope
<i>decode_config_update_envelope(config_update_envelope_bytes)</i>	Decodes config update envelope
<i>decode_config_update(config_update_bytes)</i>	Decodes update bytes in configuration
<i>decode_config_groups(config_group_map)</i>	Decodes configuration groups inside ConfigGroup
<i>decode_config_group(proto_config_group)</i>	Decodes configuration group from config protos
<i>decode_config_values(config_value_map)</i>	Decodes configuration values inside each configuration key
<i>decode_config_value(proto_config_value, key)</i>	Decodes ConfigValue from map with a given key
<i>decode_config_policies(config_policy_map)</i>	Decodes list of configuration policies
<i>decode_config_policy(proto_config_policy)</i>	Decodes config policy based on type of policy
<i>decode_implicit_meta_policy(implicit_meta_policy_bytes)</i>	Decodes implicit meta policy in a policy
<i>decode_signature_policy_envelope(signature_policy_envelope_bytes)</i>	Decodes signature policy envelope bytes
<i>decode_signature_policy(proto_signature_policy)</i>	Decodes signature policy based on field
<i>decode_MSP_principal(proto_msp_principal)</i>	Decodes MSP Principal
<i>decode_config_signature(proto_configSignature)</i>	Decodes Configuration Signature
<i>decode_fabric_MSP_config(msp_config_bytes)</i>	Decodes Fabric MSP Configuration
<i>decode_fabric_OU_identifier(FabricOUIdentifier)</i>	Decodes Fabric OU Identifier
<i>decode_fabric_Nodes_OUs(proto_node_organizationalUnits)</i>	Decodes Fabric Node OUs
<i>to_PEM_certs(buffer_array_in)</i>	Decodes String buffer input to PEM Certs
<i>decode_signing_identity_info(signing_identity_info_bytes)</i>	Decodes Signing identity information from MSP Configuration
<i>decode_key_info(key_info_bytes)</i>	Decodes Key Infor in MSP Configuration
<i>decode_crypto_config(crypto_config_bytes)</i>	Decodes Crypto Config in MSP Configuration
<i>decode_chaincode_action_payload(payload_bytes)</i>	Decodes chaincode action payload from ChaincodeAction

Continued on next page

Table 16 – continued from previous page

<code>decode_chaincode_proposal_payload(chaincode_proposal_payload_bytes)</code>	Decodes chaincode proposal payload from ChaincodeProposal
<code>decode_chaincode_endorsed_action(proto_chaincode_endorsed_action)</code>	Decodes chaincode endorsed action
<code>decode_endorsement(proto_endorsement)</code>	Decodes each endorsement
<code>decode_proposal_response_payload(proposal_response_payload_bytes)</code>	Decodes response payload in the proposal
<code>decode_chaincode_action(action_bytes)</code>	Decodes chaincode actions
<code>decode_chaincode_events(event_bytes)</code>	Decodes events in the chaincode
<code>decode_chaincode_id(proto_chaincode_id)</code>	Decodes chaincode ID information
<code>decode_readwrite_sets(rw_sets_bytes)</code>	Decodes read write sets from a given TxReadWriteSet
<code>decode_kv_rw_set(kv_bytes)</code>	Decodes Key Value Read Write Set from KV Bytes
<code>decode_kv_read(proto_kv_read)</code>	Decodes Key Value Read
<code>decode_range_query_info(proto_range_query_info)</code>	Decodes range query information from KV RW sets.
<code>decode_kv_write(proto_kv_write)</code>	Decodes key value write instance
<code>decode_response(proto_response)</code>	Decodes response containing status, message and payload
<code>decode_fabric_peers_info(peers_info_bytes)</code>	Decodes Fabric Peers Information
<code>decode_fabric_endpoints(endpoints)</code>	Decodes Fabric Endpoints

`hfc.fabric.block_decoder._logger`

class `hfc.fabric.block_decoder.BlockDecoder`

Bases: object

An object of a fully decoded protobuf message “Block”

static `decode(block_bytes)`

Constructs a JSON Object containing all decoded values from protobuf encoded *Block* bytes.

Parameters `block_bytes` (*bytes*) – Block instance

Returns Dictionary containing decoded Block instance.

static `decode_transaction(processed_tx_bytes)`

Decodes a transaction proto and constructs a deserialized object

Parameters `processed_tx_bytes` – Binary content of tx

Returns Dictionary containing tx block information

Raises **ValueError** – If data is not passed to the method

class `hfc.fabric.block_decoder.FilteredBlockDecoder`

Bases: object

An object of a fully decoded protobuf message “FilteredBlock”

static `decode(block_bytes)`

Constructs a JSON Object containing all decoded values from protobuf encoded *FilteredBlock* bytes.

Parameters `block_bytes` (*bytes*) – FilteredBlock instance

Returns Dictionary containing decoded Filtered Block instance.

`hfc.fabric.block_decoder.tx_validation_code`

`hfc.fabric.block_decoder.type_as_string`

`hfc.fabric.block_decoder.implicit_metapolicy_rule = ['ANY', 'ALL', 'MAJORITY']`

`hfc.fabric.block_decoder.policy_policy_type = ['UNKNOWN', 'SIGNATURE', 'MSP', 'IMPLICIT_ME']`

```

class hfc.fabric.block_decoder.HeaderType
    Bases: object

    HeaderType class having decodePayload and convertToString methods

    static convert_to_string (type_value)

    static decode_payload_based_on_type (proto_data, type_value)

hfc.fabric.block_decoder.decode_block_header (proto_block_header)
    Decodes the header of Block

    Parameters proto_block_header (str) – Block Header proto

    Returns Decoded BlockHeader inside Block instance.

hfc.fabric.block_decoder.decode_block_data (proto_block_data, not_proto=False)
    Decodes the data of Block.

    Parameters

    • proto_block_data (str) – Block Data proto.
    • not_proto (bool) – Boolean for if proto.

    Returns deserialized block_data (Default value = False)

hfc.fabric.block_decoder.decode_block_metadata (proto_block_metadata)
    Decodes block metadata from block

    Parameters proto_block_metadata (bytes) – Block metadata proto content

    Returns deserialized metadata contents

hfc.fabric.block_decoder.decode_block_data_envelope (proto_envelope)
    Decodes the envelope contents of Block

    Parameters proto_envelope (str) – Envelope proto

    Returns deserialized block envelope

hfc.fabric.block_decoder.decode_header (proto_header)
    Decodes the Payload header in envelope

    Parameters proto_header (str) – Envelope Payload

    Returns deserialized envelope header

hfc.fabric.block_decoder.decode_channel_header (header_bytes)
    Decodes channel header for Payload channel header

    Parameters header_bytes (str) – Bytes channel header

    Returns deserialized payload channel_header

hfc.fabric.block_decoder.timestamp_to_date (timestamp)
    Converts timestamp to current date

    Parameters timestamp – Timestamp value

    Returns String formatted date in %Y-%m-%d %H:%M:%S

hfc.fabric.block_decoder.decode_version (version_long)
    Takes version proto object and returns version

    Parameters version_long – version proto object

    Returns integer value of version_long

```

`hfc.fabric.block_decoder.decode_signature_header` (*signature_header_bytes*)
Decode signature header

Parameters `signature_header_bytes` – signature header bytes

Returns deserialized `signature_header`

`hfc.fabric.block_decoder.decode_identity` (*id_bytes*)
Decodes identity

Parameters `id_bytes` – byte of identity

Returns deserialized identity

`hfc.fabric.block_decoder.decode_metadata_signatures` (*metadata_bytes*)
Decodes metadata signature from bytes

Parameters `metadata_bytes` (*str*) – Metadata object proto

Returns deserialized Metadata blocks

`hfc.fabric.block_decoder.decode_metadata_value_signatures` (*proto_meta_signatures*)
Decodes all signatures in metadata values

Parameters `proto_meta_signatures` (*list (str)*) – List of value objects

Returns deserialized list of signatures from metadata values

`hfc.fabric.block_decoder.decode_last_config_sequence_number` (*metadata_bytes*)
Decodes last configuration and index for sequence number

Parameters `metadata_bytes` (*str*) – encoded content for sequence number

Returns deserialized dictionary of config sequence number

`hfc.fabric.block_decoder.decode_transaction_filter` (*metadata_bytes*)
Decodes transaction filter from metadata bytes

Parameters `metadata_bytes` (*str*) – Encoded list of transaction filters

Returns decoded `transaction_filter` list

`hfc.fabric.block_decoder.decode_endorser_transaction` (*trans_bytes*)
Decodes

Parameters `trans_bytes` – Serialized endorser transaction bytes

Returns deserialized dictionary of endorser transaction data

`hfc.fabric.block_decoder.decode_config_envelope` (*config_envelope_bytes*)
Decodes configuration envelope

Parameters `config_envelope_bytes` – byte of config envelope

Returns deserialized config envelope

`hfc.fabric.block_decoder.decode_config` (*proto_config*)
Decodes configuration from config envelope

Parameters `proto_config` (*bytes*) – Config value

Returns deserialized config

`hfc.fabric.block_decoder.decode_config_update_envelope` (*config_update_envelope_bytes*)
Decode config update envelope

Parameters `config_update_envelope_bytes` (*str*) – Bytes of update envelope

Returns deserialized config update envelope signatures

`hfc.fabric.block_decoder.decode_config_update` (*config_update_bytes*)
Decodes update bytes in configuration

Parameters `config_update_bytes` (*str*) – Bytes

Returns deserialized configuration update

`hfc.fabric.block_decoder.decode_config_groups` (*config_group_map*)
Decodes configuration groups inside ConfigGroup

Parameters `config_group_map` (*str*) – Serialized ConfigGroup.groups object

Returns map of configuration groups.

`hfc.fabric.block_decoder.decode_config_group` (*proto_config_group*)
Decodes configuration group from config protos

Parameters `proto_config_group` (*str*) – serialized ConfigGroup() object

Returns deserialized config_groups dictionary

`hfc.fabric.block_decoder.decode_config_values` (*config_value_map*)
Decodes configuration values inside each configuration key

Parameters `config_value_map` (*str*) – Serialized values map for each config key

Returns map of configuration values for each key

`hfc.fabric.block_decoder.decode_config_value` (*proto_config_value, key*)
Decodes ConfigValue from map with a given key

Parameters

- `proto_config_value` (*str*) – A bytes string of config_value
- `key` (*str*) – Map key for the configuration value

Returns Dictionary of configuration value deserialized

`hfc.fabric.block_decoder.decode_config_policies` (*config_policy_map*)
Decodes list of configuration policies

Parameters `config_policy_map` (*str*) – Serialized list of configuration policies

Returns deserialized map of config policies.

`hfc.fabric.block_decoder.decode_config_policy` (*proto_config_policy*)
Decodes config policy based on type of policy

Parameters `proto_config_policy` – Configuration policy bytes

Returns deserialized config_policy based on policy type.

`hfc.fabric.block_decoder.decode_implicit_meta_policy` (*implicit_meta_policy_bytes*)
Decodes implicit meta policy in a policy

Parameters `implicit_meta_policy_bytes` (*str*) – Bytes of implicit meta policy

Returns deserialized implicit_meta_policy value.

`hfc.fabric.block_decoder.decode_signature_policy_envelope` (*signature_policy_envelope_bytes*)
Decodes signature policy envelope bytes

Parameters `signature_policy_envelope_bytes` (*str*) – Serialized signature envelope

Returns deserialized signature policy envelope contents.

`hfc.fabric.block_decoder.decode_signature_policy(proto_signature_policy)`
Decodes signature policy based on field

Parameters `proto_signature_policy` – Object of SignaturePolicy()

Returns deserialized signature policy after decoding based on field.

`hfc.fabric.block_decoder.decode_MSP_principal(proto_msp_principal)`
Decodes MSP Principal

Parameters `proto_msp_principal (str)` – Bytes for MSP Principals

Returns deserialized MSP Principal based on classification.

`hfc.fabric.block_decoder.decode_config_signature(proto_configSignature)`
Decodes Configuration Signature

Parameters `proto_configSignature (str)` – ConfigSignature() object

Returns deserialized config signature after header decode.

`hfc.fabric.block_decoder.decode_fabric_MSP_config(msp_config_bytes)`
Decodes Fabric MSP Configuration

Parameters `msp_config_bytes (str)` – Serialized configuration for MSP

Returns Deserialized MSP configuration and certs.

`hfc.fabric.block_decoder.decode_fabric_OU_identifier(FabricOUIdentifier)`
Decodes Fabric OU Identifier

Parameters `FabricOUIdentifier (str)` – OU Identifier

Returns OU Identifier object.

`hfc.fabric.block_decoder.decode_fabric_Nodes_OUs(proto_node_organizational_units)`
Decodes Fabric Node OUs

Parameters `proto_node_organizational_units (str)` – OUs

Returns deserialized list of OU Identifier objects.

`hfc.fabric.block_decoder.to_PEM_certs(buffer_array_in)`
Decodes String buffer input to PEM Certs

Parameters `buffer_array_in (str)` – certificate contents buffer

Returns Concat buffer contents and returns certs

`hfc.fabric.block_decoder.decode_signing_identity_info(signing_identity_info_bytes)`
Decodes Signing identity information from MSP Configuration

Parameters `signing_identity_info_bytes (str)` – Byte string of the identity info

Returns deserialized signing identity information.

`hfc.fabric.block_decoder.decode_key_info(key_info_bytes)`
Decodes Key Infor in MSP Configuration

Parameters `key_info_bytes (str)` – Byte information containing KeyInfo

Returns deserialized key information.

`hfc.fabric.block_decoder.decode_crypto_config(crypto_config_bytes)`
Decodes Crypto Config in MSP Configuration

Parameters `crypto_config_bytes (str)` – Byte information of FabricCryptoConfig

Returns deserialized key information.

`hfc.fabric.block_decoder.decode_chaincode_action_payload(payload_bytes)`
Decodes chaincode action payload from ChaincodeAction

Parameters `payload_bytes` (*str*) – Bytes buffer of the payload

Returns deserialized payload information and action.

`hfc.fabric.block_decoder.decode_chaincode_proposal_payload(chaincode_proposal_payload_bytes)`
Decodes chaincode proposal payload from ChaincodeProposal

Parameters `chaincode_proposal_payload_bytes` (*str*) – Bytes of chaincode proposal

Returns deserialized chaincode proposal payload information

`hfc.fabric.block_decoder.decode_chaincode_endorsed_action(proto_chaincode_endorsed_action)`
Decodes chaincode endorsed action

Parameters `proto_chaincode_endorsed_action` – Object containing endorsements

Returns deserialized chaincode endorsement action.

`hfc.fabric.block_decoder.decode_endorsement(proto_endorsement)`
Decodes each endorsement

Parameters `proto_endorsement` – Object of endorsed content containing endorser & related signature

Returns deserialized endorsement content

`hfc.fabric.block_decoder.decode_proposal_response_payload(proposal_response_payload_bytes)`
Decodes response payload in the proposal

Parameters `proposal_response_payload_bytes` – Byte string of response payload

Returns deserialized proposal response payload.

`hfc.fabric.block_decoder.decode_chaincode_action(action_bytes)`
Decodes chaincode actions

Parameters `action_bytes` (*str*) – Byte buffer of the chaincode action

Returns deserialized chaincode action of results, events and response

`hfc.fabric.block_decoder.decode_chaincode_events(event_bytes)`
Decodes events in the chaincode

Parameters `event_bytes` (*str*) – Byte buffer of event content

Returns deserialized event contents.

`hfc.fabric.block_decoder.decode_chaincode_id(proto_chaincode_id)`
Decodes chaincode ID information

Parameters `proto_chaincode_id` – Object containing chaincode details

Returns deserialized chaincode ID with path, name and version.

`hfc.fabric.block_decoder.decode_readwrite_sets(rw_sets_bytes)`
Decodes read write sets from a given TxReadWriteSet

Parameters `rw_sets_bytes` (*str*) – Byte buffer of the TxReadWriteSet

Returns deserialized transaction read write set contents.

`hfc.fabric.block_decoder.decode_kv_rw_set(kv_bytes)`
Decodes Key Value Read Write Set from KV Bytes

Parameters `kv_bytes` (*str*) – Buffer of key value bytes

Returns deserialized key value read write set of reads, writes and range queries information.

`hfc.fabric.block_decoder.decode_kv_read(proto_kv_read)`

Decodes Key Value Read

Parameters `proto_kv_read` – Object of the key value with read contents

Returns deserialized key value read contents with block num and tx_num

`hfc.fabric.block_decoder.decode_range_query_info(proto_range_query_info)`

Decodes range query information from KV RW sets.

Parameters `proto_range_query_info` – Object of key value read write range queries

Returns deserialized range query information with merkle hashes.

`hfc.fabric.block_decoder.decode_kv_write(proto_kv_write)`

Decodes key value write instance

Parameters `proto_kv_write` – Object containing key value writes

Returns deserialized key value write contents and values.

`hfc.fabric.block_decoder.decode_response(proto_response)`

Decodes response containing status, message and payload

Parameters `proto_response` – Object containing proto responses

Returns deserialized response from protobuf objects

`hfc.fabric.block_decoder.decode_fabric_peers_info(peers_info_bytes)`

Decodes Fabric Peers Information

Parameters `peers_info_bytes` (*str*) – Serialized information about Peer

Returns Deserialized Peers information and certs.

`hfc.fabric.block_decoder.decode_fabric_endpoints(endpoints)`

Decodes Fabric Endpoints

Parameters `endpoints` (*str*) – Fabric Endpoints

Returns Deserialized endpoints.

`hfc.fabric.certificateAuthority`

Module Contents

Classes

`certificateAuthority(name='ca')`

An organization in the network.

Functions

<code>create_ca(name, info)</code>	Factory method to construct a ca instance
------------------------------------	---

`hfc.fabric.certificateAuthority._logger`

class `hfc.fabric.certificateAuthority.certificateAuthority` (*name='ca'*)

Bases: object

An organization in the network.

It contains several members.

init_with_bundle (*self, info*)

Init the peer with given info dict :param info: Dict including all info, e.g., :return: True or False

`hfc.fabric.certificateAuthority.create_ca` (*name, info*)

Factory method to construct a ca instance

Args: name: Name of the ca info: Info dict for initialization

Returns: an organization instance

`hfc.fabric.client`

Module Contents

Classes

<code>Client</code> (<i>net_profile=None</i>)	Main interaction handler with end user.
---	---

`hfc.fabric.client.consoleHandler`

`hfc.fabric.client._logger`

class `hfc.fabric.client.Client` (*net_profile=None*)

Bases: object

Main interaction handler with end user. Client can maintain several channels.

Parameters object –

init_with_net_profile (*self, profile_path='network.json'*)

Load the connection profile from external file to `network_info`.

Init the handlers for orgs, peers, orderers, ca nodes

Parameters `profile_path` – The connection profile file path

Returns

async init_with_discovery (*self, requestor, peer_target, channel_name=None*)

Load the connection profile from discover.

Init the handlers for orgs, peers, orderers, ca nodes

Returns

async close_grpc_channels (*self*)

Close the peers/orderers gRPC channels :return:

set_tls_client_cert_and_key (*self*, *client_key_file=None*, *client_cert_file=None*)

Set tls client certificate and key for mutual tls for all peers and orderers

Parameters

- **client_key_file** (*str*) – file path for Private key used for TLS when making client connections, defaults to None
- **client_cert_file** (*str*) – file path for X.509 certificate used for TLS when making client connections, defaults to None

Returns set success value

Return type Boolean

get_user (*self*, *org_name*, *name*)

Get a user instance. :param org_name: Name of org belongs to :param name: Name of the user :return: user instance or None

get_orderer (*self*, *name*)

Get an orderer instance with the name. :param name: Name of the orderer node. :return: The orderer instance or None.

get_peer (*self*, *name*)

Get a peer instance with the name. :param name: Name of the peer node. :return: The peer instance or None.

export_net_profile (*self*, *export_file='network_exported.json'*)

Export the current network profile into external file :param export_file: External file to save the result into :return:

get_net_info (*self*, **key_path*)

Get the info from self.network_info :param key_path: path of the key, e.g., a.b.c means info['a']['b']['c'] :return: The value, or None

property organizations (*self*)

Get the organizations in the network.

Returns organizations as dict

property orderers (*self*)

Get the orderers in the network.

Returns orderers as dict

property peers (*self*)

Get the peers instance in the network.

Returns peers as dict

property CAs (*self*)

Get the CAs in the network.

Returns CAs as dict

new_channel (*self*, *name*)

Create a channel handler instance with given name.

Parameters **name** (*str*) – The name of the channel.

Returns The inited channel.

Return type channel

get_channel (*self*, *name*)

Get a channel handler instance.

Parameters **name** (*str*) – The name of the channel.

Returns Get the channel instance with the name or None

Return type str/None

async channel_create (*self*, *orderer*, *channel_name*, *requestor*, *config_yaml=None*, *channel_profile=None*, *config_tx=None*)

Create a channel, send request to orderer, and check the response

Parameters **orderer** – Name or Orderer instance of orderer to get

genesis block from :param channel_name: Name of channel to create :param requestor: Name of creator :param config_yaml: Directory path of config yaml to be set for **FABRIC_CFG_PATH** variable :param channel_profile: Name of the channel profile defined inside config yaml file :param config_tx: Path of the configtx file of createchannel generated with configtxgen :return: True (creation succeeds) or False (creation failed)

async channel_update (*self*, *orderer*, *channel_name*, *requestor*, *config_yaml=None*, *channel_profile=None*, *config_tx=None*, *signatures=None*)

Update a channel, send request to orderer, and check the response

Parameters **orderer** – Name or Orderer instance of orderer to get

genesis block from :param channel_name: Name of channel to create :param requestor: Name of creator :param config_tx: Path of the configtx file of createchannel generated with configtxgen :return: True (creation succeeds) or False (creation failed)

async channel_join (*self*, *requestor*, *channel_name*, *peers*, *orderer*)

Join a channel. Get genesis block from orderer, then send request to peer

Parameters

- **requestor** – User to send the request
- **channel_name** – Name of channel to create
- **peers** – List of peers to join to the channel
- **orderer** – Name or Orderer instance of orderer to get

genesis block from

Returns True (creation succeeds) or False (creation failed)

async get_channel_config (*self*, *requestor*, *channel_name*, *peers*, *decode=True*)

Get configuration block for the channel

Parameters

- **requestor** – User role who issue the request
- **channel_name** – name of channel to query
- **peers** – Names or Instance of the peers to query
- **decode** – Decode the response payload

Returns A *ChaincodeQueryResponse* or *ProposalResponse*

async get_channel_config_with_orderer (*self*, *requestor*, *channel_name*, *orderer=None*)

Get configuration block for the channel with the orderer

Parameters

- **requestor** – User role who issue the request
- **channel_name** – name of channel to query
- **orderer** – Names or Instance of the orderer to query

Returns A ConfigEnvelope

extract_channel_config (*self, config_envelope*)

Extracts the protobuf ‘ConfigUpdate’ out of the ‘ConfigEnvelope’ that is produced by the configtxgen tool.

The returned object may then be signed using `sign_channel_config()` method.

Once all the signatures have been collected, the ‘ConfigUpdate’ object and the signatures may be used on `create_channel()` or `update_channel()` calls

Parameters **config_envelope** (*bytes*) – encoded bytes of the ConfigEnvelope protobuf

Returns encoded bytes of ConfigUpdate protobuf, ready to be signed

Return type bytes

sign_channel_config (*self, config, to_string=True*)

This method uses the client instance’s current signing identity to sign over the configuration bytes passed in.

Parameters

- **config** – The configuration update in bytes form.
- **tx_context** – Transaction Context
- **to_string** (*bool*) – Whether to convert the result to string, defaults to True

Returns The signature of the current user of the config bytes.

Return type config_signature(common_pb2.ConfigSignature)

channel_signconfigtx (*self, config_tx_file, requestor*)

async _create_or_update_channel (*self, request*)

Calls the orderer to start building the new channel.

Parameters **request** (*dict*) – The create channel request.

Returns OrdererResponse or an error.

Return type Response/Error

_validate_request (*self, request*)

Validate a request :param request: request to validate :return:

_create_or_update_channel_request (*self, request, have_envelope*)

Initiates the create of update channel process.

Parameters

- **request** (*dict*) – A create_update channel request.
- **have_envelope** (*Boolean*) – Signals if the requests contains a finished protobuf envelope.

Returns BroadcastResponse which includes status and info

property crypto_suite (*self*)

Get the crypto suite.

Returns The crypto_suite instance or None

property tx_context (*self*)

Get the current tx_context for the client.

Returns The tx_context object or None

Return type object/None

property state_store (*self*)

Get the KeyValue store.

Returns Return the keyValue store instance or None

Return type object/None

send_install_proposal (*self, tx_context, peers*)

Send install proposal

Parameters

- **tx_context** – transaction context
- **peers** – peers

Returns A set of proposal_response

send_instantiate_proposal (*self, tx_context, peers, channel_name*)

Send instantiate proposal

Parameters

- **tx_context** – transaction context
- **peers** – peers
- **channel_name** (*str*) – name of the channel

Returns A set of proposal_response

send_upgrade_proposal (*self, tx_context, peers, channel_name*)

Send upgrade proposal

Parameters

- **tx_context** – transaction context
- **peers** – peers
- **channel_name** (*str*) – the name of channel

Returns A set of proposal_response

generate_channel_tx (*self, channel_name, cfg_path, channel_profile*)

Creates channel configuration transaction

Parameters

- **channel_name** (*str*) – Name of the channel
- **cfg_path** (*str*) – Directory path of config yaml to be set for
- **channel_profile** (*str*) – Name of the channel profile defined inside

Returns path to tx file if success else None

Return type str/None

async chaincode_install (*self, requestor, peers, cc_path, cc_name, cc_version, packaged_cc=None, transient_map=None*)

Install chaincode to given peers by requestor role

Parameters

- **requestor** – User role who issue the request
- **peers** – List of peer name and/or Peer to install
- **cc_path** – chaincode path
- **cc_name** – chaincode name
- **cc_version** – chaincode version
- **packaged_cc** – packaged chaincode
- **transient_map** – transient map

Returns True or False

txEvent (*self, tx_id, tx_status, block_number*)

create_onCcEventArray (*self, _uuid*)

create_onCcEvent (*self, _uuid*)

async chaincode_instantiate (*self, requestor, channel_name, peers, args, cc_name, cc_version, cc_endorsement_policy=None, transient_map=None, collections_config=None, wait_for_event=False, wait_for_event_timeout=30*)

Instantiate installed chaincode to particular peer in particular channel

Parameters

- **requestor** – User role who issue the request
- **channel_name** – the name of the channel to send tx proposal
- **peers** – List of peer name and/or Peer to install
- **(list) (args)** – arguments (keys and values) for initialization
- **cc_name** – chaincode name
- **cc_version** – chaincode version
- **cc_endorsement_policy** – chaincode endorsement policy
- **transient_map** – transient map
- **collection_config** – collection configuration
- **wait_for_event** – Whether to wait for the event from each peer’s deliver filtered service signifying that the ‘invoke’ transaction has
been committed successfully
- **wait_for_event_timeout** – Time to wait for the event from each peer’s deliver filtered service signifying that the ‘invoke’
transaction has been committed successfully (default 30s)

Returns chaincode data payload

```
async chaincode_upgrade (self, requestor, channel_name, peers, cc_name, cc_version,
                          cc_endorsement_policy=None, fcn='init', args=None, transient_map=None,
                          collections_config=None, wait_for_event=False,
                          wait_for_event_timeout=30)
```

Upgrade installed chaincode to particular peer in particular channel

Parameters

- **requestor** – User role who issue the request
- **channel_name** – the name of the channel to send tx proposal
- **peers** – List of peer name and/or Peer to install
- **(list) (args)** – arguments (keys and values) for initialization
- **cc_name** – chaincode name
- **cc_version** – chaincode version
- **cc_endorsement_policy** – chaincode endorsement policy
- **fcn** – chaincode function to send
- **args** – chaincode function arguments
- **transient_map** – transient map
- **collection_config** – collection configuration
- **wait_for_event** – Whether to wait for the event from each peer’s deliver filtered service signifying that the ‘invoke’ transaction has been committed successfully
- **wait_for_event_timeout** – Time to wait for the event from each peer’s deliver filtered service signifying that the ‘invoke’ transaction has been committed successfully (default 30s)

Returns chaincode data payload

```
async chaincode_invoke (self, requestor, channel_name, peers, args,
                          cc_name, cc_type=CC_TYPE_GOLANG, fcn='invoke',
                          cc_pattern=None, transient_map=None, wait_for_event=False,
                          wait_for_event_timeout=30, grpc_broker_unavailable_retry=0,
                          grpc_broker_unavailable_retry_delay=3000,
                          raise_broker_unavailable=True)
```

Invoke chaincode for ledger update

Parameters

- **requestor** – User role who issue the request
- **channel_name** – the name of the channel to send tx proposal
- **peers** – List of peer name and/or Peer to install
- **(list) (args)** – arguments (keys and values) for initialization
- **cc_name** – chaincode name
- **cc_type** – chaincode type language

- **fcn** – chaincode function
- **cc_pattern** – chaincode event name regex
- **transient_map** – transient map
- **wait_for_event** – Whether to wait for the event from each peer’s deliver filtered service signifying that the ‘invoke’ transaction has
 been committed successfully
- **wait_for_event_timeout** – Time to wait for the event from each peer’s deliver filtered service signifying that the ‘invoke’
 transaction has been committed successfully (default 30s)
- **grpc_broker_unavailable_retry** – Number of retry if a broker is unavailable (default 0)

:param grpc_broker_unavailable_retry_delay [Delay in ms to retry] (default 3000 ms)

Parameters raise_broker_unavailable – Raise if any broker is unavailable, else always send the proposal regardless of unavailable brokers.

Returns invoke result

async chaincode_query (*self, requestor, channel_name, peers, args, cc_name, cc_type=CC_TYPE_GOLANG, fcn='query', transient_map=None*)

Query chaincode

Parameters

- **requestor** – User role who issue the request
- **channel_name** – the name of the channel to send tx proposal
- **peers** – List of peer name and/or Peer to install
- **(list) (args)** – arguments (keys and values) for initialization
- **cc_name** – chaincode name
- **cc_type** – chaincode type language
- **fcn** – chaincode function
- **transient_map** – transient map

Returns True or False

async query_channels (*self, requestor, peers, transient_map=None, decode=True*)

Queries channel name joined by a peer

Parameters

- **requestor** – User role who issue the request
- **peers** – List of peer name and/or Peer to install
- **transient_map** – transient map
- **decode** – Decode the response payload

Returns A *ChannelQueryResponse* or *ProposalResponse*

async query_info (*self, requestor, channel_name, peers, decode=True*)

Queries information of a channel

Parameters

- **requestor** – User role who issue the request
- **channel_name** – Name of channel to query
- **peers** – List of peer name and/or Peer to install
- **decode** – Decode the response payload

Returns *A BlockchainInfo or ProposalResponse*

async query_block_by_txid (*self, requestor, channel_name, peers, tx_id, decode=True*)

Queries block by tx id

Parameters

- **requestor** – User role who issue the request
- **channel_name** – Name of channel to query
- **peers** – List of peer name and/or Peer to install
- **tx_id** – Transaction ID
- **decode** – Decode the response payload

Returns *A BlockDecoder or ProposalResponse*

async query_block_by_hash (*self, requestor, channel_name, peers, block_hash, decode=True*)

Queries block by hash

Parameters

- **requestor** – User role who issue the request
- **channel_name** – Name of channel to query
- **peers** – List of peer name and/or Peer to install
- **block_hash** – Hash of a block
- **decode** – Decode the response payload

Returns *A BlockDecoder or ProposalResponse*

async query_block (*self, requestor, channel_name, peers, block_number, decode=True*)

Queries block by number

Parameters

- **requestor** – User role who issue the request
- **channel_name** – name of channel to query
- **peers** – List of peer name and/or Peer to install
- **block_number** – Number of a block
- **decode** – Decode the response payload

Returns *A BlockDecoder or ProposalResponse*

async query_transaction (*self, requestor, channel_name, peers, tx_id, decode=True*)

Queries block by number

Parameters

- **requestor** – User role who issue the request

- **channel_name** – name of channel to query
- **peers** – List of peer name and/or Peer to install
- **tx_id** – The id of the transaction
- **decode** – Decode the response payload

Returns A *BlockDecoder* or *ProposalResponse*

async query_instantiated_chaincodes (*self*, *requestor*, *channel_name*, *peers*, *transient_map=None*, *decode=True*)

Queries instantiated chaincode

Parameters

- **requestor** – User role who issue the request
- **channel_name** – name of channel to query
- **peers** – Names or Instance of the peers to query
- **transient_map** – transient map
- **decode** – Decode the response payload

Returns A *ChaincodeQueryResponse* or *ProposalResponse*

async query_installed_chaincodes (*self*, *requestor*, *peers*, *transient_map=None*, *decode=True*)

Queries installed chaincode, returns all chaincodes installed on a peer

Parameters

- **requestor** – User role who issue the request
- **peers** – Names or Instance of the peers to query
- **transient_map** – transient map
- **decode** – Decode the response payload

Returns A *ChaincodeQueryResponse* or *ProposalResponse*

async query_peers (*self*, *requestor*, *peer*, *channel=None*, *local=True*, *decode=True*)

Queries peers with discovery api

Parameters

- **requestor** – User role who issue the request
- **peer** – Name or Instance of the peer to send request
- **crypto** – crypto method to sign the request
- **deocode** – Decode the response payload

Return result a nested dict of query result

_process_discovery_membership_result (*self*, *q_members*)

`hfc.fabric.orderer`

Module Contents

Classes

<code>Orderer</code> (name='orderer', point=DEFAULT_ORDERER_ENDPOINT, tls_ca_cert_file=None, client_cert_file=None, opts=None)	end- client_key_file=None,	A orderer node in the network. It has a specific grpc channel address.
---	-------------------------------	--

`hfc.fabric.orderer.DEFAULT_ORDERER_ENDPOINT = localhost:7050`

`hfc.fabric.orderer._logger`

class `hfc.fabric.orderer.Orderer` (name='orderer', endpoint=DEFAULT_ORDERER_ENDPOINT, tls_ca_cert_file=None, client_key_file=None, client_cert_file=None, opts=None)

Bases: object

A orderer node in the network. It has a specific grpc channel address. :param object: :type object:

init_with_bundle (self, info)

Init the peer with given info dict

Parameters **info** – Dict including all info, e.g., endpoint, grpc option

Returns True/False

Return type Boolean

get_genesis_block (self, tx_context, channel_name)

get the genesis block of the channel

Returns the genesis block in success or None in fail

Return type Block/None

broadcast (self, envelope)

Send an broadcast envelope to orderer.

Parameters **envelope** – The message envelope

Returns orderer_response or exception

delivery (self, envelope, scheduler=None)

Send an delivery envelop to orderer.

Parameters

- **envelope** – The message envelope
- **scheduler** – defaults to None

Returns orderer_response or exception

get_attrs (self)

__str__ (self)

Return str(self).

property endpoint (*self*)

Return the endpoint of the orderer.

Returns endpoint

property name (*self*)

Return the name of the orderer.

Returns name of orderer

Return type str

_handle_response_stream (*self, responses*)

Handle response stream.

Parameters **responses** – responses

Returns response

Return type tuple

set_tls_client_cert_and_key (*self, client_key_file=None, client_cert_file=None*)

Set tls client's cert and key for mutual tls

Parameters

- **client_key_file** (*str*) – file path for Private key used for TLS when making client connections, defaults to None
- **client_cert_file** (*str*) – file path for X.509 certificate used for TLS when making client connections, defaults to None

Returns set success value

Return type Boolean

`hfc.fabric.organization`

Module Contents

Classes

<code>Organization</code> (name='org', state_store=None)	An organization in the network.
--	---------------------------------

Functions

<code>create_org</code> (name, info, state_store)	Factory method to construct an organization instance
---	--

`hfc.fabric.organization._logger`

class `hfc.fabric.organization.Organization` (*name='org', state_store=None*)

Bases: object

An organization in the network.

It contains several members.

init_with_bundle (*self, info*)

Init the peer with given info dict :param info: Dict including all info, e.g., endpoint, grpc option :return: True or False

get_user (*self*, *name*)

Return user instance with the name. :param name: Name of the user :return: User instance or None

`hfc.fabric.organization.create_org` (*name*, *info*, *state_store*)

Factory method to construct an organization instance :param name: Name of the organization :param info: Info dict for initialization :param state_store: State store for data cache :return: an organization instance

hfc.fabric.peer

Module Contents

Classes

`Peer`(*name*='peer', *endpoint*=DEFAULT_PEER_ENDPOINT, *point*=DEFAULT_PEER_ENDPOINT, *tls_ca_cert_file*=None, *client_key_file*=None, *client_cert_file*=None, *opts*=None) end- A peer node in the network.

Functions

`create_peer`(*endpoint*=DEFAULT_PEER_ENDPOINT, *tls_cacerts*=None, *client_key*=None, *client_cert*=None, *opts*=None) Factory method to construct a peer instance

`hfc.fabric.peer.DEFAULT_PEER_ENDPOINT` = localhost:7051

`hfc.fabric.peer._logger`

class `hfc.fabric.peer.Peer` (*name*='peer', *endpoint*=DEFAULT_PEER_ENDPOINT, *tls_ca_cert_file*=None, *client_key_file*=None, *client_cert_file*=None, *opts*=None)

Bases: object

A peer node in the network.

It has a specific gRPC channel address.

send_proposal (*self*, *proposal*)

Send an endorsement proposal to endorser

Parameters *proposal* – The endorsement proposal

Returns ProposalResponse or exception

Return type Response/Exception

send_discovery (*self*, *request*)

Send an request to discovery server

Parameters *request* – a signed request

Returns QueryResult or exception

Return type Result/Exception

init_with_bundle (*self*, *info*)

Init the peer with given info dict :param info: Dict including all info, e.g., endpoint, grpc option :return: True or False

get_attrs (*self*)

__str__ (*self*)

Return str(self).

property endpoint (*self*)

Return the endpoint of the peer.

Returns endpoint

property name (*self*)

Get the peer name

Returns The peer name

Return type str

join (*self*, *chan*)

Join a channel

Parameters *chan* – a channel instance

property channels (*self*)

delivery (*self*, *envelope*, *scheduler=None*, *filtered=True*)

Send an delivery envelop to event service.

Parameters

- **envelope** – The message envelope
- **scheduler** – defaults to None
- **filtered** (*bool*) – defaults to True

Returns orderer_response or exception

set_tls_client_cert_and_key (*self*, *client_key_file=None*, *client_cert_file=None*)

Set tls client's cert and key for mutual tls

Parameters

- **client_key_file** (*str*) – file path for Private key used for TLS when making client connections, defaults to None
- **client_cert_file** (*str*) – file path for X.509 certificate used for TLS when making client connections, defaults to None

Returns set success value

Return type bool

`hfc.fabric.peer.create_peer` (*endpoint=DEFAULT_PEER_ENDPOINT*, *tls_cacerts=None*,
client_key=None, *client_cert=None*, *opts=None*)

Factory method to construct a peer instance

Parameters

- **endpoint** – endpoint, defaults to DEFAULT_PEER_ENDPOINT
- **tls_cacerts** – pem, defaults to None

- **client_key** – pem, defaults to None
- **client_cert** – pem, defaults to None
- **opts** (*opts*) – opts, defaults to None

Returns a peer instance

hfc.fabric.user

Module Contents

Classes

<i>User</i> (name, org, state_store)	The default implementation of user.
--------------------------------------	-------------------------------------

Functions

<i>validate</i> (user)	Check the user.
<i>create_user</i> (name, org, state_store, msp_id, key_path, cert_path, crypto_suite=ecies())	Create user

hfc.fabric.user._logger

class hfc.fabric.user.**User** (*name, org, state_store*)

Bases: object

The default implementation of user.

property *name* (*self*)

Get the user name :return: The user name

property *org* (*self*)

Get the org :return: The org

property *roles* (*self*)

Get the roles :return: The roles

property *account* (*self*)

Get the account :return: The account

property *affiliation* (*self*)

Get the affiliation :return: The affiliation

property *enrollment* (*self*)

Get the enrollment

property *enrollment_secret* (*self*)

Get the enrollment_secret

property *msp_id* (*self*)

Get the msp_id

property *cryptoSuite* (*self*)

Get the cryptoSuite

is_registered (*self*)
Check if user registered

Returns boolean

is_enrolled (*self*)
Check if user enrolled

Returns boolean

_save_state (*self*)
Persistent user state.

_restore_state (*self*)
Restore user state.

get_attrs (*self*)

__str__ (*self*)
Return str(self).

hfc.fabric.user.validate (*user*)
Check the user.

Parameters **user** – A user object

Returns A validated user object

Raises **ValueError** – When user property is invalid

hfc.fabric.user.create_user (*name, org, state_store, msp_id, key_path, cert_path, crypto_suite=ecies()*)

Create user

Parameters

- **name** – user's name
- **org** – org name
- **state_store** – user state store
- **msp_id** – msp id for the user
- **crypto_suite** – the cryptoSuite used to store crypto and key store settings (Default value = ecies())
- **key_path** – identity private key path
- **cert_path** – identity public cert path

Returns a user instance

Package Contents

Classes

<i>Client</i> (<i>net_profile=None</i>)	Main interaction handler with end user.
<i>NullHandler</i> (<i>level=NOTSET</i>)	Handler instances dispatch logging events to specific destinations.

class **hfc.fabric.Client** (*net_profile=None*)

Bases: object

Main interaction handler with end user. Client can maintain several channels.

Parameters object –

init_with_net_profile (*self*, *profile_path*='network.json')
Load the connection profile from external file to network_info.

Init the handlers for orgs, peers, orderers, ca nodes

Parameters profile_path – The connection profile file path

Returns

async init_with_discovery (*self*, *requestor*, *peer_target*, *channel_name*=None)
Load the connection profile from discover.

Init the handlers for orgs, peers, orderers, ca nodes

Returns

async close_grpc_channels (*self*)
Close the peers/orderers gRPC channels :return:

set_tls_client_cert_and_key (*self*, *client_key_file*=None, *client_cert_file*=None)
Set tls client certificate and key for mutual tls for all peers and orderers

Parameters

- **client_key_file** (*str*) – file path for Private key used for TLS when making client connections, defaults to None
- **client_cert_file** (*str*) – file path for X.509 certificate used for TLS when making client connections, defaults to None

Returns set success value

Return type Boolean

get_user (*self*, *org_name*, *name*)
Get a user instance. :param org_name: Name of org belongs to :param name: Name of the user :return: user instance or None

get_orderer (*self*, *name*)
Get an orderer instance with the name. :param name: Name of the orderer node. :return: The orderer instance or None.

get_peer (*self*, *name*)
Get a peer instance with the name. :param name: Name of the peer node. :return: The peer instance or None.

export_net_profile (*self*, *export_file*='network_exported.json')
Export the current network profile into external file :param export_file: External file to save the result into :return:

get_net_info (*self*, **key_path*)
Get the info from self.network_info :param key_path: path of the key, e.g., a.b.c means info['a']['b']['c'] :return: The value, or None

property organizations (*self*)
Get the organizations in the network.

Returns organizations as dict

property orderers (*self*)

Get the orderers in the network.

Returns orderers as dict

property peers (*self*)

Get the peers instance in the network.

Returns peers as dict

property CAs (*self*)

Get the CAs in the network.

Returns CAs as dict

new_channel (*self, name*)

Create a channel handler instance with given name.

Parameters **name** (*str*) – The name of the channel.

Returns The inited channel.

Return type channel

get_channel (*self, name*)

Get a channel handler instance.

Parameters **name** (*str*) – The name of the channel.

Returns Get the channel instance with the name or None

Return type str/None

async channel_create (*self, orderer, channel_name, requestor, config_yaml=None, channel_profile=None, config_tx=None*)

Create a channel, send request to orderer, and check the response

Parameters **orderer** – Name or Orderer instance of orderer to get

genesis block from :param channel_name: Name of channel to create :param requestor: Name of creator :param config_yaml: Directory path of config yaml to be set for **FABRIC_CFG_PATH** variable :param channel_profile: Name of the channel profile defined inside config yaml file :param config_tx: Path of the configtx file of createchannel generated with configtxgen :return: True (creation succeeds) or False (creation failed)

async channel_update (*self, orderer, channel_name, requestor, config_yaml=None, channel_profile=None, config_tx=None, signatures=None*)

Update a channel, send request to orderer, and check the response

Parameters **orderer** – Name or Orderer instance of orderer to get

genesis block from :param channel_name: Name of channel to create :param requestor: Name of creator :param config_tx: Path of the configtx file of createchannel generated with configtxgen :return: True (creation succeeds) or False (creation failed)

async channel_join (*self, requestor, channel_name, peers, orderer*)

Join a channel. Get genesis block from orderer, then send request to peer

Parameters

- **requestor** – User to send the request
- **channel_name** – Name of channel to create
- **peers** – List of peers to join to the channel
- **orderer** – Name or Orderer instance of orderer to get

genesis block from

Returns True (creation succeeds) or False (creation failed)

async get_channel_config (*self, requestor, channel_name, peers, decode=True*)

Get configuration block for the channel

Parameters

- **requestor** – User role who issue the request
- **channel_name** – name of channel to query
- **peers** – Names or Instance of the peers to query
- **decode** – Decode the response payload

Returns A *ChaincodeQueryResponse* or *ProposalResponse*

async get_channel_config_with_orderer (*self, requestor, channel_name, orderer=None*)

Get configuration block for the channel with the orderer

Parameters

- **requestor** – User role who issue the request
- **channel_name** – name of channel to query
- **orderer** – Names or Instance of the orderer to query

Returns A *ConfigEnvelope*

extract_channel_config (*self, config_envelope*)

Extracts the protobuf ‘ConfigUpdate’ out of the ‘ConfigEnvelope’ that is produced by the configtxgen tool.

The returned object may then be signed using `sign_channel_config()` method.

Once all the signatures have been collected, the ‘ConfigUpdate’ object and the signatures may be used on `create_channel()` or `update_channel()` calls

Parameters **config_envelope** (*bytes*) – encoded bytes of the *ConfigEnvelope* protobuf

Returns encoded bytes of *ConfigUpdate* protobuf, ready to be signed

Return type bytes

sign_channel_config (*self, config, to_string=True*)

This method uses the client instance’s current signing identity to sign over the configuration bytes passed in.

Parameters

- **config** – The configuration update in bytes form.
- **tx_context** – Transaction Context
- **to_string** (*bool*) – Whether to convert the result to string, defaults to True

Returns The signature of the current user of the config bytes.

Return type `config_signature(common_pb2.ConfigSignature)`

channel_signconfigtx (*self, config_tx_file, requestor*)

async _create_or_update_channel (*self, request*)

Calls the orderer to start building the new channel.

Parameters **request** (*dict*) – The create channel request.

Returns OrdererResponse or an error.

Return type Response/Error

_validate_request (*self, request*)

Validate a request :param request: request to validate :return:

_create_or_update_channel_request (*self, request, have_envelope*)

Initiates the create or update channel process.

Parameters

- **request** (*dict*) – A create_update channel request.
- **have_envelope** (*Boolean*) – Signals if the requests contains a finished protobuf envelope.

Returns BroadcastResponse which includes status and info

property crypto_suite (*self*)

Get the crypto suite.

Returns The crypto_suite instance or None

property tx_context (*self*)

Get the current tx_context for the client.

Returns The tx_context object or None

Return type object/None

property state_store (*self*)

Get the KeyValue store.

Returns Return the keyValue store instance or None

Return type object/None

send_install_proposal (*self, tx_context, peers*)

Send install proposal

Parameters

- **tx_context** – transaction context
- **peers** – peers

Returns A set of proposal_response

send_instantiate_proposal (*self, tx_context, peers, channel_name*)

Send instantiate proposal

Parameters

- **tx_context** – transaction context
- **peers** – peers
- **channel_name** (*str*) – name of the channel

Returns A set of proposal_response

send_upgrade_proposal (*self*, *tx_context*, *peers*, *channel_name*)

Send upgrade proposal

Parameters

- **tx_context** – transaction context
- **peers** – peers
- **channel_name** (*str*) – the name of channel

Returns A set of proposal_response

generate_channel_tx (*self*, *channel_name*, *cfg_path*, *channel_profile*)

Creates channel configuration transaction

Parameters

- **channel_name** (*str*) – Name of the channel
- **cfg_path** (*str*) – Directory path of config yaml to be set for
- **channel_profile** (*str*) – Name of the channel profile defined inside

Returns path to tx file if success else None

Return type str/None

async chaincode_install (*self*, *requestor*, *peers*, *cc_path*, *cc_name*, *cc_version*, *packaged_cc=None*, *transient_map=None*)

Install chaincode to given peers by requestor role

Parameters

- **requestor** – User role who issue the request
- **peers** – List of peer name and/or Peer to install
- **cc_path** – chaincode path
- **cc_name** – chaincode name
- **cc_version** – chaincode version
- **packaged_cc** – packaged chaincode
- **transient_map** – transient map

Returns True or False

txEvent (*self*, *tx_id*, *tx_status*, *block_number*)

create_onCcEventArray (*self*, *_uuid*)

create_onCcEvent (*self*, *_uuid*)

async chaincode_instantiate (*self*, *requestor*, *channel_name*, *peers*, *args*, *cc_name*, *cc_version*, *cc_endorsement_policy=None*, *transient_map=None*, *collections_config=None*, *wait_for_event=False*, *wait_for_event_timeout=30*)

Instantiate installed chaincode to particular peer in particular channel

Parameters

- **requestor** – User role who issue the request
- **channel_name** – the name of the channel to send tx proposal

- **peers** – List of peer name and/or Peer to install
- **(list) (args)** – arguments (keys and values) for initialization
- **cc_name** – chaincode name
- **cc_version** – chaincode version
- **cc_endorsement_policy** – chaincode endorsement policy
- **transient_map** – transient map
- **collection_config** – collection configuration
- **wait_for_event** – Whether to wait for the event from each peer’s deliver filtered service signifying that the ‘invoke’ transaction has
been committed successfully
- **wait_for_event_timeout** – Time to wait for the event from each peer’s deliver filtered service signifying that the ‘invoke’
transaction has been committed successfully (default 30s)

Returns chaincode data payload

```
async chaincode_upgrade(self, requestor, channel_name, peers, cc_name, cc_version,  
                        cc_endorsement_policy=None, fcn='init', args=None, tran-  
sient_map=None, collections_config=None, wait_for_event=False,  
                        wait_for_event_timeout=30)
```

Upgrade installed chaincode to particular peer in particular channel

Parameters

- **requestor** – User role who issue the request
- **channel_name** – the name of the channel to send tx proposal
- **peers** – List of peer name and/or Peer to install
- **(list) (args)** – arguments (keys and values) for initialization
- **cc_name** – chaincode name
- **cc_version** – chaincode version
- **cc_endorsement_policy** – chaincode endorsement policy
- **fcn** – chaincode function to send
- **args** – chaincode function arguments
- **transient_map** – transient map
- **collection_config** – collection configuration
- **wait_for_event** – Whether to wait for the event from each peer’s deliver filtered service signifying that the ‘invoke’ transaction has
been committed successfully
- **wait_for_event_timeout** – Time to wait for the event from each peer’s deliver filtered service signifying that the ‘invoke’
transaction has been committed successfully (default 30s)

Returns chaincode data payload

```
async chaincode_invoke (self, requestor, channel_name, peers, args,
                        cc_name, cc_type=CC_TYPE_GOLANG, fcn='invoke',
                        cc_pattern=None, transient_map=None, wait_for_event=False,
                        wait_for_event_timeout=30, grpc_broker_unavailable_retry=0,
                        grpc_broker_unavailable_retry_delay=3000,
                        raise_broker_unavailable=True)
```

Invoke chaincode for ledger update

Parameters

- **requestor** – User role who issue the request
- **channel_name** – the name of the channel to send tx proposal
- **peers** – List of peer name and/or Peer to install
- **(list) (args)** – arguments (keys and values) for initialization
- **cc_name** – chaincode name
- **cc_type** – chaincode type language
- **fcn** – chaincode function
- **cc_pattern** – chaincode event name regex
- **transient_map** – transient map
- **wait_for_event** – Whether to wait for the event from each peer’s deliver filtered service signifying that the ‘invoke’ transaction has been committed successfully
- **wait_for_event_timeout** – Time to wait for the event from each peer’s deliver filtered service signifying that the ‘invoke’ transaction has been committed successfully (default 30s)
- **grpc_broker_unavailable_retry** – Number of retry if a broker is unavailable (default 0)

:param grpc_broker_unavailable_retry_delay [Delay in ms to retry] (default 3000 ms)

Parameters raise_broker_unavailable – Raise if any broker is unavailable, else always send the proposal regardless of unavailable brokers.

Returns invoke result

```
async chaincode_query (self, requestor, channel_name, peers, args, cc_name,
                        cc_type=CC_TYPE_GOLANG, fcn='query', transient_map=None)
```

Query chaincode

Parameters

- **requestor** – User role who issue the request
- **channel_name** – the name of the channel to send tx proposal
- **peers** – List of peer name and/or Peer to install
- **(list) (args)** – arguments (keys and values) for initialization
- **cc_name** – chaincode name
- **cc_type** – chaincode type language

- **fcn** – chaincode function
- **transient_map** – transient map

Returns True or False

async query_channels (*self, requestor, peers, transient_map=None, decode=True*)

Queries channel name joined by a peer

Parameters

- **requestor** – User role who issue the request
- **peers** – List of peer name and/or Peer to install
- **transient_map** – transient map
- **decode** – Decode the response payload

Returns A *ChannelQueryResponse* or *ProposalResponse*

async query_info (*self, requestor, channel_name, peers, decode=True*)

Queries information of a channel

Parameters

- **requestor** – User role who issue the request
- **channel_name** – Name of channel to query
- **peers** – List of peer name and/or Peer to install
- **deocode** – Decode the response payload

Returns A *BlockchainInfo* or *ProposalResponse*

async query_block_by_txid (*self, requestor, channel_name, peers, tx_id, decode=True*)

Queries block by tx id

Parameters

- **requestor** – User role who issue the request
- **channel_name** – Name of channel to query
- **peers** – List of peer name and/or Peer to install
- **tx_id** – Transaction ID
- **deocode** – Decode the response payload

Returns A *BlockDecoder* or *ProposalResponse*

async query_block_by_hash (*self, requestor, channel_name, peers, block_hash, decode=True*)

Queries block by hash

Parameters

- **requestor** – User role who issue the request
- **channel_name** – Name of channel to query
- **peers** – List of peer name and/or Peer to install
- **block_hash** – Hash of a block
- **deocode** – Decode the response payload

Returns A *BlockDecoder* or *ProposalResponse*

async query_block (*self, requestor, channel_name, peers, block_number, decode=True*)
 Queries block by number

Parameters

- **requestor** – User role who issue the request
- **channel_name** – name of channel to query
- **peers** – List of peer name and/or Peer to install
- **block_number** – Number of a block
- **decode** – Decode the response payload

Returns *A BlockDecoder or ProposalResponse*

async query_transaction (*self, requestor, channel_name, peers, tx_id, decode=True*)
 Queries block by number

Parameters

- **requestor** – User role who issue the request
- **channel_name** – name of channel to query
- **peers** – List of peer name and/or Peer to install
- **tx_id** – The id of the transaction
- **decode** – Decode the response payload

Returns *A BlockDecoder or ProposalResponse*

async query_instantiated_chaincodes (*self, requestor, channel_name, peers, transient_map=None, decode=True*)

Queries instantiated chaincode

Parameters

- **requestor** – User role who issue the request
- **channel_name** – name of channel to query
- **peers** – Names or Instance of the peers to query
- **transient_map** – transient map
- **decode** – Decode the response payload

Returns *A ChaincodeQueryResponse or ProposalResponse*

async query_installed_chaincodes (*self, requestor, peers, transient_map=None, decode=True*)

Queries installed chaincode, returns all chaincodes installed on a peer

Parameters

- **requestor** – User role who issue the request
- **peers** – Names or Instance of the peers to query
- **transient_map** – transient map
- **decode** – Decode the response payload

Returns *A ChaincodeQueryResponse or ProposalResponse*

async query_peers (*self, requestor, peer, channel=None, local=True, decode=True*)
 Queries peers with discovery api

Parameters

- **requestor** – User role who issue the request
- **peer** – Name or Instance of the peer to send request
- **crypto** – crypto method to sign the request
- **decode** – Decode the response payload

Return result a nested dict of query result

`_process_discovery_membership_result` (*self, q_members*)

class `hfc.fabric.NullHandler` (*level=NOTSET*)

Bases: `logging.Handler`

Handler instances dispatch logging events to specific destinations.

The base handler class. Acts as a placeholder which defines the Handler interface. Handlers can optionally use Formatter instances to format records as desired. By default, no formatter is specified; in this case, the ‘raw’ message as determined by `record.message` is logged.

emit (*self, record*)

Do whatever it takes to actually log the specified logging record.

This version is intended to be implemented by subclasses and so raises a `NotImplementedError`.

`hfc.fabric_ca`

Submodules

`hfc.fabric_ca.affiliationService`

Module Contents**Classes**

`AffiliationService`(*client*)

`hfc.fabric_ca.affiliationService._logger`

class `hfc.fabric_ca.affiliationService.AffiliationService` (*client*)

Bases: `object`

create (*self, registrar, name, caname="", force=False*)

Create a new affiliation. The caller must have `hf.AffiliationMgr` authority.

If any of the parent affiliations do not exist and ‘force’ is true, create all parent affiliations also.

Parameters

- **registrar** – Required. The identity of the registrar (i.e. who is performing the registration)
- **name** – The affiliation path to create
- **caname** – Optional. Name of the CA to send the request to within the Fabric CA server (Default value = ‘’)

- **force** – Optional. (Default value = False)

Raises

- **RequestException** – errors in requests.exceptions
- **ValueError** – Failed response, json parse error, args missing

Returns result

getOne (*self, affiliation, registrar*)

List a specific affiliation at or below the caller's affinity. The caller must have hf.AffiliationMgr authority.

Parameters

- **affiliation** – The affiliation path to be queried.
- **registrar** – Required. The identity of the registrar (i.e. who is performing the registration)

Raises

- **RequestException** – errors in requests.exceptions
- **ValueError** – Failed response, json parse error, args missing

Returns result

getAll (*self, registrar*)

List all affiliations equal to and below the caller's affiliation. The caller must have hf.AffiliationMgr authority.

Parameters **registrar** – Required. The identity of the registrar (i.e. who is performing the registration)

Returns result

Raises

- **RequestException** – errors in requests.exceptions
- **ValueError** – Failed response, json parse error, args missing

delete (*self, affiliation, registrar, force=False*)

Delete an affiliation. The caller must have hf.AffiliationMgr authority. Ca server must have `cfg.affiliations.allowremove: true`

If force is true and there are any child affiliations or any identities are associated with this affiliation or child affiliations, these

identities and child affiliations

will be deleted; otherwise, an error is returned.

Parameters

- **affiliation** – affiliation
- **registrar** – Required. The identity of the registrar (i.e. who is performing the registration)
- **force** – (Default value = False)

Returns result

Raises

- **RequestException** – errors in requests.exceptions

- **ValueError** – Failed response, json parse error, args missing

update (*self, affiliation, registrar, name, caname="", force=False*)

Rename an affiliation. The caller must have hf.AffiliationMgr authority.

If any identities are associated with this affiliation, ‘force’ is true causes these identities’ affiliations to be renamed; otherwise, an error is returned.

Parameters

- **affiliation** – The affiliation path to be updated.
- **registrar** – Required. The identity of the registrar (i.e. who is performing the registration)
- **name** – The affiliation path to create
- **caname** – Optional. Name of the CA to send the request to within the Fabric CA server (Default value = ‘’)
- **force** – Optional. (Default value = False)

Returns result

Raises

- **RequestException** – errors in requests.exceptions
- **ValueError** – Failed response, json parse error, args missing

hfc.fabric_ca.caservice

Module Contents

Classes

<code>Enrollment</code> (private_key, enrollmentCert, caC- ertChain=None, service=None)	Class represents enrollment.
<code>CAClient</code> (target=DEFAULT_CA_ENDPOINT, ca_certs_path=None, base_url=DEFAULT_CA_BASE_URL, ca_name='', cryptoPrimitives=ecies())	Client for communicating with the Fabric CA APIs.
<code>CAService</code> (target=DEFAULT_CA_ENDPOINT, ca_certs_path=None, crypto=ecies(), ca_name='')	This is a ca server delegate.

Functions

<code>ca_service</code> (target=DEFAULT_CA_ENDPOINT, ca_certs_path=None, crypto=ecies(), ca_name='')	Create ca service
---	-------------------

`hfc.fabric_ca.caservice.DEFAULT_CA_ENDPOINT = http://localhost:7054`

`hfc.fabric_ca.caservice.DEFAULT_CA_BASE_URL = /api/v1/`

`hfc.fabric_ca.caservice._logger`

`hfc.fabric_ca.caservice.reasons = [[1, 'unspecified'], [2, 'keycompromise'], [3, 'cacompro`

```
class hfc.fabric_ca.caservice.Enrollment (private_key, enrollmentCert, caCertChain=None,  
                                           service=None)
```

Bases: object

Class represents enrollment.

```
property private_key (self)
```

Get private key

Returns private key

```
property cert (self)
```

Get cert

Returns cert

```
property caCert (self)
```

Get caCert

Returns caCert

```
get_attrs (self)
```

```
register (self, enrollmentID, enrollmentSecret=None, role=None, affiliation=None, maxEnroll-  
         ments=1, attrs=None)
```

```
revoke (self, enrollmentID=None, aki=None, serial=None, reason=None, genCrl=False)
```

```
generateCRL (self, revokedBefore=None, revokedAfter=None, expireBefore=None, ex-  
             pireAfter=None)
```

```
__str__ (self)
```

Return str(self).

```
class hfc.fabric_ca.caservice.CAClient (target=DEFAULT_CA_ENDPOINT,  
                                         ca_certs_path=None,  
                                         base_url=DEFAULT_CA_BASE_URL, ca_name="",  
                                         cryptoPrimitives=ecies())
```

Bases: object

Client for communicating with the Fabric CA APIs.

```
generateAuthToken (self, req, registrar)
```

Generate authorization token required for accessing fabric-ca APIs

Parameters

- **req** (*dict*) – request body
- **registrar** – Required. The identity of the registrar

(i.e. who is performing the request) :type registrar: Enrollment :return: auth token

```
_send_ca_post (self, path, **param)
```

Send a post request to the ca service

Parameters

- **path** – sub path after the base_url
- ****param** – post request params

Returns the response body in json

```
_send_ca_get (self, path, **param)
```

Send a get request to the ca service

Parameters

- **path** – sub path after the base_url
- ****param** – get request params

Returns the response body in json

`_send_ca_delete (self, path, **param)`

Send a delete request to the ca service

Parameters

- **path** – sub path after the base_url
- ****param** – delete request params

Returns the response body in json

`_send_ca_update (self, path, **param)`

Send a update request to the ca service

Parameters

- **path** – sub path after the base_url
- ****param** – update request params

Returns the response body in json

`get_cainfo (self)`

Query the ca service information.

Returns The base64 encoded CA PEM file content for the caname

`enroll (self, enrollment_id, enrollment_secret, csr, profile="", attr_reqs=None)`

`register (self, req, registrar)`

`reenroll (self, req, registrar)`

`revoke (self, req, registrar)`

`generateCRL (self, req, registrar)`

`newIdentityService (self)`

`newAffiliationService (self)`

`newCertificateService (self)`

```
class hfc.fabric_ca.caservice.CAService (target=DEFAULT_CA_ENDPOINT,  
                                         ca_certs_path=None,          crypto=ecies(),  
                                         ca_name="")
```

Bases: object

This is a ca server delegate.

`enroll (self, enrollment_id, enrollment_secret, csr=None, profile="", attr_reqs=None)`

Enroll a registered user in order to receive a signed X509 certificate

Parameters

- **enrollment_id** (*str*) – The registered ID to use for enrollment
- **enrollment_secret** (*str*) – The secret associated with the enrollment ID
- **profile** (*str*) – The profile name. Specify the ‘tls’ profile for a TLS certificate; otherwise, an enrollment certificate is issued. (Default value = ‘’)

- **csr** (*str*) – Optional. PEM-encoded PKCS#10 Certificate Signing Request. The message sent from client side to Fabric-ca for the digital identity certificate. (Default value = None)
- **attr_reqs** (*list*) – An array of AttributeRequest

Returns PEM-encoded X509 certificate (Default value = None)

Raises

- **RequestException** – errors in requests.exceptions
- **ValueError** – Failed response, json parse error, args missing

reenroll (*self, currentUser, attr_reqs=None*)

Re-enroll the member in cases such as the existing enrollment certificate is about to expire, or it has been compromised

Parameters

- **currentUser** (*Enrollment*) – The identity of the current user that holds the existing enrollment certificate
- **attr_reqs** (*list*) – Optional. An array of AttributeRequest that indicate attributes to be included in the certificate

Returns PEM-encoded X509 certificate (Default value = None)

Raises

- **RequestException** – errors in requests.exceptions
- **ValueError** – Failed response, json parse error, args missing

register (*self, enrollmentID, enrollmentSecret, role, affiliation, maxEnrollments, attrs, registrar*)

Register a user in order to receive a secret

Parameters

- **registrar** (*Enrollment*) – The registrar
- **enrollmentID** (*str*) – enrollmentID ID which will be used for enrollment
- **enrollmentSecret** (*str*) – enrollmentSecret Optional enrollment secret to set for the registered user. If not provided, the server will generate one. When not including, use a null for this parameter.
- **role** (*str*) – Optional type of role for this user. When not including, use a null for this parameter.
- **affiliation** (*str*) – Affiliation with which this user will be associated
- **maxEnrollments** (*number*) – The maximum number of times the user is permitted to enroll
- **attrs** – Array of key/value attributes to assign to the user

:return The enrollment secret to use when this user enrolls

Raises

- **RequestException** – errors in requests.exceptions

- **ValueError** – Failed response, json parse error, args missing

revoke (*self*, *enrollmentID*, *aki*, *serial*, *reason*, *genctrl*, *registrar*)

Revoke an existing certificate (enrollment certificate or

transaction certificate), or revoke all certificates issued to an

enrollment id. If revoking a particular certificate, then both the

Authority Key Identifier and serial number are required. If

revoking by enrollment id, then all future requests to enroll this id will be rejected.

Parameters

- **registrar** (*Enrollment*) – The registrar
- **enrollmentID** (*str*) – enrollmentID ID to revoke
- **aki** (*str*) – Authority Key Identifier string, hex encoded, for the specific certificate to revoke
- **serial** (*str*) – Serial number string, hex encoded, for the specific certificate to revoke
- **reason** (*str*) – The reason for revocation. See <https://godoc.org/golang.org/x/crypto/ocsp> for valid values
- **genctrl** (*bool*) – GenCRL specifies whether to generate a CRL

Returns The revocation results

Raises

- **RequestException** – errors in requests.exceptions
- **ValueError** – Failed response, json parse error, args missing

generateCRL (*self*, *revokedBefore*, *revokedAfter*, *expireBefore*, *expireAfter*, *registrar*)

Generate CRL

Parameters

- **revokedBefore** – Include certificates that were revoked before this UTC timestamp (in RFC3339 format) in the CRL
- **revokedAfter** – Include certificates that were revoked after this UTC timestamp (in RFC3339 format) in the CRL
- **expireBefore** – Include revoked certificates that expire before this UTC timestamp (in RFC3339 format) in the CRL
- **expireAfter** – Include revoked certificates that expire after this UTC timestamp (in RFC3339 format) in the CRL
- **registrar** – registrar

Returns The Certificate Revocation List (CRL)

newIdentityService (*self*)

newAffiliationService (*self*)

newCertificateService (*self*)


```
hfc.fabric_ca.caservice.ca_service(target=DEFAULT_CA_ENDPOINT,
                                   ca_certs_path=None, crypto=ecies(), ca_name="")
```

Create ca service

Parameters

- **target** – url (Default value = DEFAULT_CA_ENDPOINT)
- **ca_certs_path** – certs path (Default value = None)
- **crypto** – crypto (Default value = ecies())
- **ca_name** – CA name

Returns ca service instance (Default value = “”)

```
hfc.fabric_ca.certificateService
```

Module Contents

Classes

```
CertificateService(client)
```

```
hfc.fabric_ca.certificateService._logger
```

```
class hfc.fabric_ca.certificateService.CertificateService(client)
```

Bases: object

```
getCertificates(self, registrar, id=None, aki=None, serial=None, revoked_start=None,
                 revoked_end=None, expired_start=None, expired_end=None, notexpired=None,
                 notrevoked=None, ca=None)
```

The caller will be able to view certificates that it owns. In addition, if the caller has **hf.Registrar.Roles** or **hf.Revoker** attribute, it will be able to view certificates for identities that have

affiliations equal to or below the caller’s affiliation.

Parameters

- **registrar** – Required. The identity of the registrar (i.e. who is performing the revocation) signing certificate, hash algorithm and signature algorithm
- **id** – The enrollment ID that uniquely identifies an identity (Default value = None)
- **aki** – Authority Key Identifier string, hex encoded, for the specific certificate (Default value = None)
- **serial** – The serial number for a certificate (Default value = None)
- **revoked_start** – Get revoked certificates starting at the specified time, either as timestamp (RFC3339 format) or duration (-30d) (Default value = None)
- **revoked_end** – Get revoked certificates before the specified time, either as timestamp * (RFC3339 format) or duration (-15d) (Default value = None)
- **expired_start** – Get expired certificates starting at the specified time, either as timestamp (RFC3339 format) or duration (-30d) (Default value = None)

- **expired_end** – Get expired certificates before the specified time, either as timestamp (RFC3339 format) or duration (-15d) (Default value = None)
- **notexpired** – Don't return expired certificates (Default value = None)
- **notrevoked** – Don't return revoked certificates (Default value = None)
- **ca** – The name of the CA to direct this request to within the server, or the default CA if not specified (Default value = None)

Returns result

Raises

- **RequestException** – errors in requests.exceptions
- **ValueError** – Failed response, json parse error, args missing

`hfc.fabric_ca.identityService`

Module Contents

Classes

IdentityService(client)

`hfc.fabric_ca.identityService._logger`

class `hfc.fabric_ca.identityService.IdentityService` (*client*)

Bases: object

create (*self, registrar, enrollmentID, enrollmentSecret=None, role=None, affiliation=None, maxEnrollments=1, attrs=None*)

Create a new identity with the Fabric CA server. An enrollment secret is returned which can then be used, along with the enrollment ID, to enroll a new identity. The caller must have *hf.Registrar* authority.

Parameters

- **registrar** – registrar
- **enrollmentID** – enrollmentID ID which will be used for enrollment
- **enrollmentSecret** – enrollmentSecret Optional enrollment secret to set for the registered user.

If not provided, the server will generate one. When not including, use a null for this parameter.(Default value = None) :param role: Optional type of role for this user. When not including, use a null for this parameter. (Default value = None) :param affiliation: Affiliation with which this user will be associated (Default value = None) :param maxEnrollments: The maximum number of times the user is permitted to enroll (Default value = 1) :param attrs: Array of key/value attributes to assign to the user. (Default value = None) :return: The enrollment secret to use when this user enrolls :raises RequestException: errors in requests.exceptions :raises ValueError: Failed response, json parse error, args missing

getOne (*self, enrollmentID, registrar*)

getAll (*self, registrar*)

delete (*self, enrollmentID, registrar, force=False*)

update (*self*, *enrollmentID*, *registrar*, *type=None*, *affiliation=None*, *maxEnrollments=None*, *attrs=None*, *enrollmentSecret=None*, *caname=None*)

Package Contents

Classes

<i>CAClient</i> (<i>target=DEFAULT_CA_ENDPOINT</i> , <i>ca_certs_path=None</i> , <i>base_url=DEFAULT_CA_BASE_URL</i> , <i>ca_name=''</i> , <i>cryptoPrimitives=ecies()</i>)	Client for communicating with the Fabric CA APIs.
<i>NullHandler</i> (<i>level=NOTSET</i>)	Handler instances dispatch logging events to specific destinations.

```
class hfc.fabric_ca.CAClient (target=DEFAULT_CA_ENDPOINT,           ca_certs_path=None,
                               base_url=DEFAULT_CA_BASE_URL,     ca_name="",      crypto-
                               Primitives=ecies())
```

Bases: object

Client for communicating with the Fabric CA APIs.

generateAuthToken (*self*, *req*, *registrar*)

Generate authorization token required for accessing fabric-ca APIs

Parameters

- **req** (*dict*) – request body
- **registrar** – Required. The identity of the registrar

(i.e. who is performing the request) :type registrar: Enrollment :return: auth token

_send_ca_post (*self*, *path*, ***param*)

Send a post request to the ca service

Parameters

- **path** – sub path after the base_url
- ****param** – post request params

Returns the response body in json

_send_ca_get (*self*, *path*, ***param*)

Send a get request to the ca service

Parameters

- **path** – sub path after the base_url
- ****param** – get request params

Returns the response body in json

_send_ca_delete (*self*, *path*, ***param*)

Send a delete request to the ca service

Parameters

- **path** – sub path after the base_url
- ****param** – delete request params

Returns the response body in json

`_send_ca_update` (*self*, *path*, ***param*)
Send a update request to the ca service

Parameters

- **path** – sub path after the base_url
- ****param** – update request params

Returns the response body in json

`get_cainfo` (*self*)
Query the ca service information.

Returns The base64 encoded CA PEM file content for the caname

`enroll` (*self*, *enrollment_id*, *enrollment_secret*, *csr*, *profile=""*, *attr_reqs=None*)

`register` (*self*, *req*, *registrar*)

`reenroll` (*self*, *req*, *registrar*)

`revoke` (*self*, *req*, *registrar*)

`generateCRL` (*self*, *req*, *registrar*)

`newIdentityService` (*self*)

`newAffiliationService` (*self*)

`newCertificateService` (*self*)

class `hfc.fabric_ca.NullHandler` (*level=NOTSET*)
Bases: `logging.Handler`

Handler instances dispatch logging events to specific destinations.

The base handler class. Acts as a placeholder which defines the Handler interface. Handlers can optionally use Formatter instances to format records as desired. By default, no formatter is specified; in this case, the 'raw' message as determined by `record.message` is logged.

`emit` (*self*, *record*)
Do whatever it takes to actually log the specified logging record.

This version is intended to be implemented by subclasses and so raises a `NotImplementedError`.

`hfc.fabric_network`

Submodules

`hfc.fabric_network.contract`

Module Contents

Classes

<code>Contract(network, cc_name, gateway)</code>	Represents a smart contract (chaincode) instance in a network.
--	--

`hfc.fabric_network.contract.consoleHandler`

`hfc.fabric_network.contract._logger`

class `hfc.fabric_network.contract.Contract` (*network, cc_name, gateway*)

Bases: object

Represents a smart contract (chaincode) instance in a network. Applications should get a Contract instance using the network's `get_contract` method. :return: an instance of Contract

get_network (*self*)

get_cc_name (*self*)

get_options (*self*)

async submit_transaction (*self, name, args, requestor*)

Submit a transaction to the ledger. The transaction function will be evaluated on the list of peers discovered and then submitted to the ordering service for committing to the ledger.

async evaluate_transaction (*self, name, args, requestor*)

Evaluate a transaction function and return its results. The transaction function will be evaluated on the endorsing peers but the responses will not be sent to the ordering service and hence will not be committed to the ledger. This is used for querying the world state.

`hfc.fabric_network.couchdbwalletstore`

Module Contents

Classes

<code>CouchDBWalletStore(dbName, config='http://localhost:5984')</code>	con-	CouchDBWalletStore stores the identities of users and admins
---	------	--

class `hfc.fabric_network.couchdbwalletstore.CouchDBWalletStore` (*dbName, config='http://localhost:5984'*)

Bases: object

CouchDBWalletStore stores the identities of users and admins in a CouchDB with given config ie. it contains the Private Key and Enrollment Certificate

exists (*self, enrollment_id*)

Returns whether or not the creds of a user with a given user_id exists in the wallet

Parameters `enrollment_id` – enrollment id

Returns True or False

remove (*self, enrollment_id*)

deletes identities of user with given enrollment_id

Parameters `enrollment_id` – enrollment id

Returns

put (*self, enrollment_id, user_enrollment*)
Saves the particular Identity in the wallet

Parameters

- **enrollment_id** – enrollment id
- **user_enrollment** – Enrollment object

Returns

create_user (*self, enrollment_id, org, msp_id, state_store=None*)

Returns an instance of a user whose identity is stored in the CouchDBWallet

Parameters

- **enrollment_id** – enrollment id
- **org** – organization
- **msp_id** – MSP id
- **state_store** – (Default value = None)

Returns a validated user instance

`hfc.fabric_network.gateway`

Module Contents

Classes

Gateway()

The gateway peer provides the connection point for an application to access the Fabric network.

`hfc.fabric_network.gateway.consoleHandler`

`hfc.fabric_network.gateway._logger`

class `hfc.fabric_network.gateway.Gateway`

Bases: object

The gateway peer provides the connection point for an application to access the Fabric network. It can then be connected to a fabric network using the path to network profile.

mergeOptions (*self, currentOptions, additionalOptions*)

Merge additional options to current options

Parameters

- **currentOptions** – current options
- **additionalOptions** – additional options to be merged

Returns result

async connect (*self, net_profile, options*)

Connect to the Gateway with a connection profile and connection options. :param net_profile: Path to the Connection Profile :param options: Options such as wallet identity and user identity :return:

get_current_identity (*self*)

Returns The current identity being used in the gateway.

get_client (*self*)

Returns Client instance.

get_options (*self*)

Returns the options being used.

disconnect (*self*)

Clean up and disconnect this Gateway connection

async get_network (*self, network_name, requestor*)

Returns an object representing a network :param Name of the channel :param requestor: User role who issue the request :return: Network instance

hfc.fabric_network.inmemorywalletstore

Module Contents

Classes

InMemoryWalletStore()

InMemoryWalletStore stores the identities of users and admins

class hfc.fabric_network.inmemorywalletstore.**InMemoryWalletStore**

Bases: object

InMemoryWalletStore stores the identities of users and admins in memory

exists (*self, enrollment_id*)

Returns whether or not the credentials of a user with a given enrollment_id exists in the wallet

Parameters **enrollment_id** – enrollment id

Returns True or False

remove (*self, enrollment_id*)

Deletes identities of users with the given user_id

Parameters **enrollment_id** – enrollment id

Returns

put (*self, enrollment_id, user_enrollment*)

Saves the particular Identity in the wallet

Parameters

- **enrollment_id** – enrollment id
- **user_enrollment** – Enrollment object

Returns

create_user (*self, enrollment_id, org, msp_id, state_store=None*)

Returns an instance of a user whose **identity** is stored in the InMemoryWallet

Parameters

- **enrollment_id** – enrollment id
- **org** – organization
- **msp_id** – MSP id
- **state_store** – (Default value = None)

Returns a validated user object

`hfc.fabric_network.network`

Module Contents

Classes

<code>Network(gateway, channel)</code>	A Network represents the set of peers in a Fabric network.
--	--

`hfc.fabric_network.network.consoleHandler`

`hfc.fabric_network.network._logger`

class `hfc.fabric_network.network.Network(gateway, channel)`

Bases: object

A Network represents the set of peers in a Fabric network. Applications should get a Network instance using the gateway's `getNetwork` method.

async `__init_internal_channel(self, discovery)`

Initialize the channel if it hasn't been done :param discovery: must include requestor :return:

async `_initialize(self, discover=None)`

Initialize this network instance :param discover: :return:

get_contract `(self, chaincode_id)`

`hfc.fabric_network.wallet`

Module Contents

Classes

<code>FileSystemWallet(path=os.getcwd() + '/tmp/hfc-kvs')</code>	FileSystemWallet stores the identities of users and admins
--	--

<code>Identity(enrollment_id, user)</code>	Class represents a tuple containing
--	-------------------------------------

class `hfc.fabric_network.wallet.FileSystemWallet(path=os.getcwd() + '/tmp/hfc-kvs')`

Bases: object

FileSystemWallet stores the identities of users and admins ie. it contains the Private Key and Enrollment Certificate

exists (*self*, *enrollment_id*)

Returns whether or not the credentials of a user with a given **user_id** exists in the wallet

Parameters **enrollment_id** – enrollment id

Returns True or False

remove (*self*, *enrollment_id*)

Deletes identities of users with the given **user_id**

Parameters **enrollment_id** – enrollment id

Returns

create_user (*self*, *enrollment_id*, *org*, *msp_id*, *state_store=None*)

Returns an instance of a user whose **identity** is stored in the FileSystemWallet

Parameters

- **enrollment_id** – enrollment id
- **org** – organization
- **msp_id** – MSP id
- **state_store** – state store (Default value = None)

Returns a user instance

class `hfc.fabric_network.wallet.Identity` (*enrollment_id*, *user*)

Bases: object

Class represents a tuple containing 1) enrollment_id 2) Enrollment Certificate of user 3) Private Key of user

CreateIdentity (*self*, *Wallet*)

Saves the particular Identity in the wallet

Parameters **Wallet** –

Returns

`hfc.protos`

Subpackages

`hfc.protos.common`

Submodules

`hfc.protos.common.collection_pb2`

Module Contents

`hfc.protos.common.collection_pb2._b`

`hfc.protos.common.collection_pb2._sym_db`

`hfc.protos.common.collection_pb2.DESRIPTOR`

hfc.protos.common.collection_pb2._COLLECTIONCONFIGPACKAGE
hfc.protos.common.collection_pb2._COLLECTIONCONFIG
hfc.protos.common.collection_pb2._STATICCOLLECTIONCONFIG
hfc.protos.common.collection_pb2._COLLECTIONPOLICYCONFIG
hfc.protos.common.collection_pb2._COLLECTIONCRITERIA
hfc.protos.common.collection_pb2.message_type
hfc.protos.common.collection_pb2.message_type
hfc.protos.common.collection_pb2.containing_oneof
hfc.protos.common.collection_pb2.message_type
hfc.protos.common.collection_pb2.message_type
hfc.protos.common.collection_pb2.containing_oneof
hfc.protos.common.collection_pb2.CollectionConfigPackage
hfc.protos.common.collection_pb2.CollectionConfig
hfc.protos.common.collection_pb2.StaticCollectionConfig
hfc.protos.common.collection_pb2.CollectionPolicyConfig
hfc.protos.common.collection_pb2.CollectionCriteria
hfc.protos.common.collection_pb2._options

hfc.protos.common.collection_pb2_grpc

hfc.protos.common.common_pb2

Module Contents

hfc.protos.common.common_pb2._b
hfc.protos.common.common_pb2._sym_db
hfc.protos.common.common_pb2.DESRIPTOR
hfc.protos.common.common_pb2._STATUS
hfc.protos.common.common_pb2.Status
hfc.protos.common.common_pb2._HEADERTYPE
hfc.protos.common.common_pb2.HeaderType
hfc.protos.common.common_pb2._BLOCKMETADATAINDEX
hfc.protos.common.common_pb2.BlockMetadataIndex
hfc.protos.common.common_pb2.UNKNOWN = 0
hfc.protos.common.common_pb2.SUCCESS = 200
hfc.protos.common.common_pb2.BAD_REQUEST = 400
hfc.protos.common.common_pb2.FORBIDDEN = 403
hfc.protos.common.common_pb2.NOT_FOUND = 404

```
hfc.protos.common.common_pb2.REQUEST_ENTITY_TOO_LARGE = 413
hfc.protos.common.common_pb2.INTERNAL_SERVER_ERROR = 500
hfc.protos.common.common_pb2.NOT_IMPLEMENTED = 501
hfc.protos.common.common_pb2.SERVICE_UNAVAILABLE = 503
hfc.protos.common.common_pb2.MESSAGE = 0
hfc.protos.common.common_pb2.CONFIG = 1
hfc.protos.common.common_pb2.CONFIG_UPDATE = 2
hfc.protos.common.common_pb2.ENDORSER_TRANSACTION = 3
hfc.protos.common.common_pb2.ORDERER_TRANSACTION = 4
hfc.protos.common.common_pb2.DELIVER_SEEK_INFO = 5
hfc.protos.common.common_pb2.CHAINCODE_PACKAGE = 6
hfc.protos.common.common_pb2.PEER_ADMIN_OPERATION = 8
hfc.protos.common.common_pb2.TOKEN_TRANSACTION = 9
hfc.protos.common.common_pb2.SIGNATURES = 0
hfc.protos.common.common_pb2.LAST_CONFIG = 1
hfc.protos.common.common_pb2.TRANSACTIONS_FILTER = 2
hfc.protos.common.common_pb2.ORDERER = 3
hfc.protos.common.common_pb2._LASTCONFIG
hfc.protos.common.common_pb2._METADATA
hfc.protos.common.common_pb2._METADATASIGNATURE
hfc.protos.common.common_pb2._HEADER
hfc.protos.common.common_pb2._CHANNELHEADER
hfc.protos.common.common_pb2._SIGNATUREHEADER
hfc.protos.common.common_pb2._PAYLOAD
hfc.protos.common.common_pb2._ENVELOPE
hfc.protos.common.common_pb2._BLOCK
hfc.protos.common.common_pb2._BLOCKHEADER
hfc.protos.common.common_pb2._BLOCKDATA
hfc.protos.common.common_pb2._BLOCKMETADATA
hfc.protos.common.common_pb2.message_type
hfc.protos.common.common_pb2.message_type
hfc.protos.common.common_pb2.message_type
hfc.protos.common.common_pb2.message_type
hfc.protos.common.common_pb2.message_type
hfc.protos.common.common_pb2.message_type
hfc.protos.common.common_pb2.LastConfig
```

`hfc.protos.common.common_pb2.Metadata`
`hfc.protos.common.common_pb2.MetadataSignature`
`hfc.protos.common.common_pb2.Header`
`hfc.protos.common.common_pb2.ChannelHeader`
`hfc.protos.common.common_pb2.SignatureHeader`
`hfc.protos.common.common_pb2.Payload`
`hfc.protos.common.common_pb2.Envelope`
`hfc.protos.common.common_pb2.Block`
`hfc.protos.common.common_pb2.BlockHeader`
`hfc.protos.common.common_pb2.BlockData`
`hfc.protos.common.common_pb2.BlockMetadata`
`hfc.protos.common.common_pb2._options`

`hfc.protos.common.common_pb2_grpc`

`hfc.protos.common.configtx_pb2`

Module Contents

`hfc.protos.common.configtx_pb2._b`
`hfc.protos.common.configtx_pb2._sym_db`
`hfc.protos.common.configtx_pb2.DESRIPTOR`
`hfc.protos.common.configtx_pb2._CONFIGENVELOPE`
`hfc.protos.common.configtx_pb2._CONFIGGROUPSCHEMA_GROUPSENTRY`
`hfc.protos.common.configtx_pb2._CONFIGGROUPSCHEMA_VALUESENTRY`
`hfc.protos.common.configtx_pb2._CONFIGGROUPSCHEMA_POLICIESENTRY`
`hfc.protos.common.configtx_pb2._CONFIGGROUPSCHEMA`
`hfc.protos.common.configtx_pb2._CONFIGVALUESCHEMA`
`hfc.protos.common.configtx_pb2._CONFIGPOLICYSCHEMA`
`hfc.protos.common.configtx_pb2._CONFIG`
`hfc.protos.common.configtx_pb2._CONFIGUPDATEENVELOPE`
`hfc.protos.common.configtx_pb2._CONFIGUPDATE_ISOLATEDDATAENTRY`
`hfc.protos.common.configtx_pb2._CONFIGUPDATE`
`hfc.protos.common.configtx_pb2._CONFIGGROUP_GROUPSENTRY`
`hfc.protos.common.configtx_pb2._CONFIGGROUP_VALUESENTRY`
`hfc.protos.common.configtx_pb2._CONFIGGROUP_POLICIESENTRY`
`hfc.protos.common.configtx_pb2._CONFIGGROUP`
`hfc.protos.common.configtx_pb2._CONFIGVALUE`

hfc.protos.common.configtx_pb2._CONFIGPOLICY
hfc.protos.common.configtx_pb2._CONFIGSIGNATURE
hfc.protos.common.configtx_pb2.message_type
hfc.protos.common.configtx_pb2.message_type
hfc.protos.common.configtx_pb2.message_type
hfc.protos.common.configtx_pb2.containing_type
hfc.protos.common.configtx_pb2.message_type
hfc.protos.common.configtx_pb2.containing_type
hfc.protos.common.configtx_pb2.message_type
hfc.protos.common.configtx_pb2.containing_type
hfc.protos.common.configtx_pb2.message_type
hfc.protos.common.configtx_pb2.message_type
hfc.protos.common.configtx_pb2.message_type
hfc.protos.common.configtx_pb2.message_type
hfc.protos.common.configtx_pb2.message_type
hfc.protos.common.configtx_pb2.containing_type
hfc.protos.common.configtx_pb2.message_type
hfc.protos.common.configtx_pb2.message_type
hfc.protos.common.configtx_pb2.message_type
hfc.protos.common.configtx_pb2.message_type
hfc.protos.common.configtx_pb2.containing_type
hfc.protos.common.configtx_pb2.message_type
hfc.protos.common.configtx_pb2.containing_type
hfc.protos.common.configtx_pb2.message_type
hfc.protos.common.configtx_pb2.containing_type
hfc.protos.common.configtx_pb2.message_type
hfc.protos.common.configtx_pb2.message_type
hfc.protos.common.configtx_pb2.message_type
hfc.protos.common.configtx_pb2.message_type
hfc.protos.common.configtx_pb2.message_type
hfc.protos.common.configtx_pb2.ConfigEnvelope
hfc.protos.common.configtx_pb2.ConfigGroupSchema
hfc.protos.common.configtx_pb2.ConfigValueSchema
hfc.protos.common.configtx_pb2.ConfigPolicySchema
hfc.protos.common.configtx_pb2.Config
hfc.protos.common.configtx_pb2.ConfigUpdateEnvelope
hfc.protos.common.configtx_pb2.ConfigUpdate

hfc.protos.common.configtx_pb2.**ConfigGroup**
hfc.protos.common.configtx_pb2.**ConfigValue**
hfc.protos.common.configtx_pb2.**ConfigPolicy**
hfc.protos.common.configtx_pb2.**ConfigSignature**
hfc.protos.common.configtx_pb2.**__options**
hfc.protos.common.configtx_pb2.**__options**
hfc.protos.common.configtx_pb2.**__options**
hfc.protos.common.configtx_pb2.**__options**
hfc.protos.common.configtx_pb2.**__options**
hfc.protos.common.configtx_pb2.**__options**
hfc.protos.common.configtx_pb2.**__options**
hfc.protos.common.configtx_pb2.**__options**

hfc.protos.common.configtx_pb2_grpc

hfc.protos.common.configuration_pb2

Module Contents

hfc.protos.common.configuration_pb2.**__b**
hfc.protos.common.configuration_pb2.**__sym_db**
hfc.protos.common.configuration_pb2.**DESCRIPTOR**
hfc.protos.common.configuration_pb2.**__HASHINGALGORITHM**
hfc.protos.common.configuration_pb2.**__BLOCKDATAHASHINGSTRUCTURE**
hfc.protos.common.configuration_pb2.**__ORDERERADDRESSES**
hfc.protos.common.configuration_pb2.**__CONSORTIUM**
hfc.protos.common.configuration_pb2.**__CAPABILITIES_CAPABILITIESENTRY**
hfc.protos.common.configuration_pb2.**__CAPABILITIES**
hfc.protos.common.configuration_pb2.**__CAPABILITY**
hfc.protos.common.configuration_pb2.**message_type**
hfc.protos.common.configuration_pb2.**containing_type**
hfc.protos.common.configuration_pb2.**message_type**
hfc.protos.common.configuration_pb2.**HashingAlgorithm**
hfc.protos.common.configuration_pb2.**BlockDataHashingStructure**
hfc.protos.common.configuration_pb2.**OrdererAddresses**
hfc.protos.common.configuration_pb2.**Consortium**
hfc.protos.common.configuration_pb2.**Capabilities**
hfc.protos.common.configuration_pb2.**Capability**

hfc.protos.common.configuration_pb2._options

hfc.protos.common.configuration_pb2._options

hfc.protos.common.configuration_pb2_grpc

hfc.protos.common.ledger_pb2

Module Contents

hfc.protos.common.ledger_pb2._b

hfc.protos.common.ledger_pb2._sym_db

hfc.protos.common.ledger_pb2.DESRIPTOR

hfc.protos.common.ledger_pb2._BLOCKCHAININFO

hfc.protos.common.ledger_pb2.BlockchainInfo

hfc.protos.common.ledger_pb2._options

hfc.protos.common.ledger_pb2_grpc

hfc.protos.common.policies_pb2

Module Contents

hfc.protos.common.policies_pb2._b

hfc.protos.common.policies_pb2._sym_db

hfc.protos.common.policies_pb2.DESRIPTOR

hfc.protos.common.policies_pb2._POLICY_POLICYTYPE

hfc.protos.common.policies_pb2._IMPLICITMETAPOLICY_RULE

hfc.protos.common.policies_pb2._POLICY

hfc.protos.common.policies_pb2._SIGNATUREPOLICYENVELOPE

hfc.protos.common.policies_pb2._SIGNATUREPOLICY_NOUTOF

hfc.protos.common.policies_pb2._SIGNATUREPOLICY

hfc.protos.common.policies_pb2._IMPLICITMETAPOLICY

hfc.protos.common.policies_pb2.containing_type

hfc.protos.common.policies_pb2.message_type

hfc.protos.common.policies_pb2.message_type

hfc.protos.common.policies_pb2.message_type

hfc.protos.common.policies_pb2.containing_type

hfc.protos.common.policies_pb2.message_type

hfc.protos.common.policies_pb2.containing_oneof

hfc.protos.common.policies_pb2.containing_oneof
hfc.protos.common.policies_pb2.enum_type
hfc.protos.common.policies_pb2.containing_type
hfc.protos.common.policies_pb2.Policy
hfc.protos.common.policies_pb2.SignaturePolicyEnvelope
hfc.protos.common.policies_pb2.SignaturePolicy
hfc.protos.common.policies_pb2.ImplicitMetaPolicy
hfc.protos.common.policies_pb2._options

hfc.protos.common.policies_pb2_grpc

hfc.protos.discovery

Submodules

hfc.protos.discovery.protocol_pb2

Module Contents

hfc.protos.discovery.protocol_pb2._b
hfc.protos.discovery.protocol_pb2._sym_db
hfc.protos.discovery.protocol_pb2.DESRIPTOR
hfc.protos.discovery.protocol_pb2._SIGNEDREQUEST
hfc.protos.discovery.protocol_pb2._REQUEST
hfc.protos.discovery.protocol_pb2._RESPONSE
hfc.protos.discovery.protocol_pb2._AUTHINFO
hfc.protos.discovery.protocol_pb2._QUERY
hfc.protos.discovery.protocol_pb2._QUERYRESULT
hfc.protos.discovery.protocol_pb2._CONFIGQUERY
hfc.protos.discovery.protocol_pb2._CONFIGRESULT_MSPENTRY
hfc.protos.discovery.protocol_pb2._CONFIGRESULT_ORDERERENTRY
hfc.protos.discovery.protocol_pb2._CONFIGRESULT
hfc.protos.discovery.protocol_pb2._PEERMEMBERSHIPQUERY
hfc.protos.discovery.protocol_pb2._PEERMEMBERSHIPRESULT_PEERSBYORENTRY
hfc.protos.discovery.protocol_pb2._PEERMEMBERSHIPRESULT
hfc.protos.discovery.protocol_pb2._CHAINCODEQUERY
hfc.protos.discovery.protocol_pb2._CHAINCODEINTEREST
hfc.protos.discovery.protocol_pb2._CHAINCODECALL
hfc.protos.discovery.protocol_pb2._CHAINCODEQUERYRESULT

hfc.protos.discovery.protocol_pb2._LOCALPEERQUERY
hfc.protos.discovery.protocol_pb2._ENDORSEMENTDESCRIPTOR_ENDORSERSBYGROUPENTRY
hfc.protos.discovery.protocol_pb2._ENDORSEMENTDESCRIPTOR
hfc.protos.discovery.protocol_pb2._LAYOUT_QUANTITIESBYGROUPENTRY
hfc.protos.discovery.protocol_pb2._LAYOUT
hfc.protos.discovery.protocol_pb2._PEERS
hfc.protos.discovery.protocol_pb2._PEER
hfc.protos.discovery.protocol_pb2._ERROR
hfc.protos.discovery.protocol_pb2._ENDPOINTS
hfc.protos.discovery.protocol_pb2._ENDPOINT
hfc.protos.discovery.protocol_pb2.message_type
hfc.protos.discovery.protocol_pb2.message_type
hfc.protos.discovery.protocol_pb2.message_type
hfc.protos.discovery.protocol_pb2.message_type
hfc.protos.discovery.protocol_pb2.message_type
hfc.protos.discovery.protocol_pb2.message_type
hfc.protos.discovery.protocol_pb2.message_type
hfc.protos.discovery.protocol_pb2.containing_oneof
hfc.protos.discovery.protocol_pb2.containing_oneof
hfc.protos.discovery.protocol_pb2.containing_oneof
hfc.protos.discovery.protocol_pb2.containing_oneof
hfc.protos.discovery.protocol_pb2.message_type
hfc.protos.discovery.protocol_pb2.message_type
hfc.protos.discovery.protocol_pb2.message_type
hfc.protos.discovery.protocol_pb2.message_type
hfc.protos.discovery.protocol_pb2.containing_oneof
hfc.protos.discovery.protocol_pb2.containing_oneof
hfc.protos.discovery.protocol_pb2.containing_oneof
hfc.protos.discovery.protocol_pb2.containing_oneof
hfc.protos.discovery.protocol_pb2.message_type
hfc.protos.discovery.protocol_pb2.containing_type
hfc.protos.discovery.protocol_pb2.message_type
hfc.protos.discovery.protocol_pb2.containing_type
hfc.protos.discovery.protocol_pb2.message_type
hfc.protos.discovery.protocol_pb2.message_type
hfc.protos.discovery.protocol_pb2.message_type

hfc.protos.discovery.protocol_pb2.**message_type**
hfc.protos.discovery.protocol_pb2.**containing_type**
hfc.protos.discovery.protocol_pb2.**message_type**
hfc.protos.discovery.protocol_pb2.**message_type**
hfc.protos.discovery.protocol_pb2.**message_type**
hfc.protos.discovery.protocol_pb2.**message_type**
hfc.protos.discovery.protocol_pb2.**message_type**
hfc.protos.discovery.protocol_pb2.**containing_type**
hfc.protos.discovery.protocol_pb2.**message_type**
hfc.protos.discovery.protocol_pb2.**message_type**
hfc.protos.discovery.protocol_pb2.**containing_type**
hfc.protos.discovery.protocol_pb2.**message_type**
hfc.protos.discovery.protocol_pb2.**message_type**
hfc.protos.discovery.protocol_pb2.**message_type**
hfc.protos.discovery.protocol_pb2.**message_type**
hfc.protos.discovery.protocol_pb2.**message_type**
hfc.protos.discovery.protocol_pb2.**SignedRequest**
hfc.protos.discovery.protocol_pb2.**Request**
hfc.protos.discovery.protocol_pb2.**Response**
hfc.protos.discovery.protocol_pb2.**AuthInfo**
hfc.protos.discovery.protocol_pb2.**Query**
hfc.protos.discovery.protocol_pb2.**QueryResult**
hfc.protos.discovery.protocol_pb2.**ConfigQuery**
hfc.protos.discovery.protocol_pb2.**ConfigResult**
hfc.protos.discovery.protocol_pb2.**PeerMembershipQuery**
hfc.protos.discovery.protocol_pb2.**PeerMembershipResult**
hfc.protos.discovery.protocol_pb2.**ChaincodeQuery**
hfc.protos.discovery.protocol_pb2.**ChaincodeInterest**
hfc.protos.discovery.protocol_pb2.**ChaincodeCall**
hfc.protos.discovery.protocol_pb2.**ChaincodeQueryResult**
hfc.protos.discovery.protocol_pb2.**LocalPeerQuery**
hfc.protos.discovery.protocol_pb2.**EndorsementDescriptor**
hfc.protos.discovery.protocol_pb2.**Layout**
hfc.protos.discovery.protocol_pb2.**Peers**
hfc.protos.discovery.protocol_pb2.**Peer**
hfc.protos.discovery.protocol_pb2.**Error**

`hfc.protos.discovery.protocol_pb2.Endpoints`
`hfc.protos.discovery.protocol_pb2.Endpoint`
`hfc.protos.discovery.protocol_pb2._options`
`hfc.protos.discovery.protocol_pb2._options`
`hfc.protos.discovery.protocol_pb2._options`
`hfc.protos.discovery.protocol_pb2._options`
`hfc.protos.discovery.protocol_pb2._options`
`hfc.protos.discovery.protocol_pb2._options`
`hfc.protos.discovery.protocol_pb2._DISCOVERY`

`hfc.protos.discovery.protocol_pb2_grpc`

Module Contents

Classes

<code>DiscoveryStub(channel)</code>	Discovery defines a service that serves information about the fabric network
<code>DiscoveryServicer()</code>	Discovery defines a service that serves information about the fabric network

Functions

`add_DiscoveryServicer_to_server(servicer, server)`

class `hfc.protos.discovery.protocol_pb2_grpc.DiscoveryStub(channel)`

Bases: object

Discovery defines a service that serves information about the fabric network like which peers, orderers, chain-codes, etc.

class `hfc.protos.discovery.protocol_pb2_grpc.DiscoveryServicer`

Bases: object

Discovery defines a service that serves information about the fabric network like which peers, orderers, chain-codes, etc.

Discover (*self, request, context*)

Discover receives a signed request, and returns a response.

`hfc.protos.discovery.protocol_pb2_grpc.add_DiscoveryServicer_to_server(servicer, server)`

`hfc.protos.gossip`

Submodules

`hfc.protos.gossip.message_pb2`

Module Contents

`hfc.protos.gossip.message_pb2._b`
`hfc.protos.gossip.message_pb2._sym_db`
`hfc.protos.gossip.message_pb2.DESRIPTOR`
`hfc.protos.gossip.message_pb2._PULLMSGTYPE`
`hfc.protos.gossip.message_pb2.PullMsgType`
`hfc.protos.gossip.message_pb2.UNDEFINED = 0`
`hfc.protos.gossip.message_pb2.BLOCK_MSG = 1`
`hfc.protos.gossip.message_pb2.IDENTITY_MSG = 2`
`hfc.protos.gossip.message_pb2._GOSSIPMESSAGE_TAG`
`hfc.protos.gossip.message_pb2._ENVELOPE`
`hfc.protos.gossip.message_pb2._SECRETENVELOPE`
`hfc.protos.gossip.message_pb2._SECRET`
`hfc.protos.gossip.message_pb2._GOSSIPMESSAGE`
`hfc.protos.gossip.message_pb2._STATEINFO`
`hfc.protos.gossip.message_pb2._PROPERTIES`
`hfc.protos.gossip.message_pb2._STATEINFOSNAPSHOT`
`hfc.protos.gossip.message_pb2._STATEINFOPULLREQUEST`
`hfc.protos.gossip.message_pb2._CONNESTABLISH`
`hfc.protos.gossip.message_pb2._PEERIDENTITY`
`hfc.protos.gossip.message_pb2._DATAREQUEST`
`hfc.protos.gossip.message_pb2._GOSSIPHELLO`
`hfc.protos.gossip.message_pb2._DATAUPDATE`
`hfc.protos.gossip.message_pb2._DATADIGEST`
`hfc.protos.gossip.message_pb2._DATAMESSAGE`
`hfc.protos.gossip.message_pb2._PRIVATEDATAMESSAGE`
`hfc.protos.gossip.message_pb2._PAYLOAD`
`hfc.protos.gossip.message_pb2._PRIVATEPAYLOAD`
`hfc.protos.gossip.message_pb2._ALIVEMESSAGE`
`hfc.protos.gossip.message_pb2._LEADERSHIPMESSAGE`
`hfc.protos.gossip.message_pb2._PEERTIME`

hfc.protos.gossip.message_pb2.**message_type**
hfc.protos.gossip.message_pb2.**message_type**
hfc.protos.gossip.message_pb2.**message_type**
hfc.protos.gossip.message_pb2.**message_type**
hfc.protos.gossip.message_pb2.**message_type**
hfc.protos.gossip.message_pb2.**message_type**
hfc.protos.gossip.message_pb2.**message_type**
hfc.protos.gossip.message_pb2.**message_type**
hfc.protos.gossip.message_pb2.**Envelope**
hfc.protos.gossip.message_pb2.**SecretEnvelope**
hfc.protos.gossip.message_pb2.**Secret**
hfc.protos.gossip.message_pb2.**GossipMessage**
hfc.protos.gossip.message_pb2.**StateInfo**
hfc.protos.gossip.message_pb2.**Properties**
hfc.protos.gossip.message_pb2.**StateInfoSnapshot**
hfc.protos.gossip.message_pb2.**StateInfoPullRequest**
hfc.protos.gossip.message_pb2.**ConnEstablish**
hfc.protos.gossip.message_pb2.**PeerIdentity**
hfc.protos.gossip.message_pb2.**DataRequest**
hfc.protos.gossip.message_pb2.**GossipHello**
hfc.protos.gossip.message_pb2.**DataUpdate**
hfc.protos.gossip.message_pb2.**DataDigest**
hfc.protos.gossip.message_pb2.**DataMessage**
hfc.protos.gossip.message_pb2.**PrivateDataMessage**
hfc.protos.gossip.message_pb2.**Payload**
hfc.protos.gossip.message_pb2.**PrivatePayload**
hfc.protos.gossip.message_pb2.**AliveMessage**
hfc.protos.gossip.message_pb2.**LeadershipMessage**
hfc.protos.gossip.message_pb2.**PeerTime**
hfc.protos.gossip.message_pb2.**MembershipRequest**
hfc.protos.gossip.message_pb2.**MembershipResponse**
hfc.protos.gossip.message_pb2.**Member**
hfc.protos.gossip.message_pb2.**Empty**
hfc.protos.gossip.message_pb2.**RemoteStateRequest**
hfc.protos.gossip.message_pb2.**RemoteStateResponse**

hfc.protos.gossip.message_pb2.**RemotePvtDataRequest**
hfc.protos.gossip.message_pb2.**PvtDataDigest**
hfc.protos.gossip.message_pb2.**RemotePvtDataResponse**
hfc.protos.gossip.message_pb2.**PvtDataElement**
hfc.protos.gossip.message_pb2.**PvtDataPayload**
hfc.protos.gossip.message_pb2.**Acknowledgement**
hfc.protos.gossip.message_pb2.**Chaincode**
hfc.protos.gossip.message_pb2.**_options**
hfc.protos.gossip.message_pb2.**_GOSSIP**

hfc.protos.gossip.message_pb2_grpc

Module Contents

Classes

<i>GossipStub</i> (channel)	Gossip
<i>GossipServicer</i> ()	Gossip

Functions

add_GossipServicer_to_server(servicer,
server)

class hfc.protos.gossip.message_pb2_grpc.**GossipStub** (*channel*)
Bases: object

Gossip

class hfc.protos.gossip.message_pb2_grpc.**GossipServicer**
Bases: object

Gossip

GossipStream (*self, request_iterator, context*)

GossipStream is the gRPC stream used for sending and receiving messages

Ping (*self, request, context*)

Ping is used to probe a remote peer's aliveness

hfc.protos.gossip.message_pb2_grpc.**add_GossipServicer_to_server** (*servicer,*
server)

`hfc.protos.idemix.idemix_pb2.ECP`
`hfc.protos.idemix.idemix_pb2.ECP2`
`hfc.protos.idemix.idemix_pb2.IssuerPublicKey`
`hfc.protos.idemix.idemix_pb2.IssuerKey`
`hfc.protos.idemix.idemix_pb2.Credential`
`hfc.protos.idemix.idemix_pb2.CredRequest`
`hfc.protos.idemix.idemix_pb2.Signature`
`hfc.protos.idemix.idemix_pb2.NonRevocationProof`
`hfc.protos.idemix.idemix_pb2.NymSignature`
`hfc.protos.idemix.idemix_pb2.CredentialRevocationInformation`
`hfc.protos.idemix.idemix_pb2._options`

`hfc.protos.idemix.idemix_pb2_grpc`

`hfc.protos.ledger`

Subpackages

`hfc.protos.ledger.queryresult`

Submodules

`hfc.protos.ledger.queryresult.kv_query_result_pb2`

Module Contents

`hfc.protos.ledger.queryresult.kv_query_result_pb2._b`
`hfc.protos.ledger.queryresult.kv_query_result_pb2._sym_db`
`hfc.protos.ledger.queryresult.kv_query_result_pb2.DESRIPTOR`
`hfc.protos.ledger.queryresult.kv_query_result_pb2._KV`
`hfc.protos.ledger.queryresult.kv_query_result_pb2._KEYMODIFICATION`
`hfc.protos.ledger.queryresult.kv_query_result_pb2.message_type`
`hfc.protos.ledger.queryresult.kv_query_result_pb2.KV`
`hfc.protos.ledger.queryresult.kv_query_result_pb2.KeyModification`
`hfc.protos.ledger.queryresult.kv_query_result_pb2.has_options = True`
`hfc.protos.ledger.queryresult.kv_query_result_pb2._options`

`hfc.protos.ledger.queryresult.kv_query_result_pb2_grpc`

`hfc.protos.ledger.rwset`

Subpackages

`hfc.protos.ledger.rwset.kvrwset`

Submodules

`hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2`

Module Contents

`hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2._b`
`hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2._sym_db`
`hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2.DESRIPTOR`
`hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2._KVRWSET`
`hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2._HASHEDRWSET`
`hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2._KVREAD`
`hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2._KVWRITE`
`hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2._KVREADHASH`
`hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2._KVWRITEHASH`
`hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2._VERSION`
`hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2._RANGEQUERYINFO`
`hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2._QUERYREADS`
`hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2._QUERYREADSMERKLESUMMARY`
`hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2.message_type`
`hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2.message_type`
`hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2.message_type`
`hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2.message_type`
`hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2.message_type`
`hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2.message_type`
`hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2.message_type`
`hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2.message_type`
`hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2.message_type`
`hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2.message_type`
`hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2.containing_oneof`
`hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2.containing_oneof`
`hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2.message_type`

`hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2.KVRWSet`
`hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2.HashedRWSet`
`hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2.KVRead`
`hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2.KVWrite`
`hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2.KVReadHash`
`hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2.KVWriteHash`
`hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2.Version`
`hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2.RangeQueryInfo`
`hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2.QueryReads`
`hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2.QueryReadsMerkleSummary`
`hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2.has_options = True`
`hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2._options`

`hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2_grpc`

Submodules

`hfc.protos.ledger.rwset.rwset_pb2`

Module Contents

`hfc.protos.ledger.rwset.rwset_pb2._b`
`hfc.protos.ledger.rwset.rwset_pb2._sym_db`
`hfc.protos.ledger.rwset.rwset_pb2.DESRIPTOR`
`hfc.protos.ledger.rwset.rwset_pb2._TXREADWRITESET_DATAMODEL`
`hfc.protos.ledger.rwset.rwset_pb2._TXREADWRITESET`
`hfc.protos.ledger.rwset.rwset_pb2._NSREADWRITESET`
`hfc.protos.ledger.rwset.rwset_pb2._COLLECTIONHASHEDREADWRITESET`
`hfc.protos.ledger.rwset.rwset_pb2._TXPVTREADWRITESET`
`hfc.protos.ledger.rwset.rwset_pb2._NSPVTREADWRITESET`
`hfc.protos.ledger.rwset.rwset_pb2._COLLECTIONPVTREADWRITESET`
`hfc.protos.ledger.rwset.rwset_pb2.enum_type`
`hfc.protos.ledger.rwset.rwset_pb2.message_type`
`hfc.protos.ledger.rwset.rwset_pb2.containing_type`
`hfc.protos.ledger.rwset.rwset_pb2.message_type`
`hfc.protos.ledger.rwset.rwset_pb2.enum_type`
`hfc.protos.ledger.rwset.rwset_pb2.message_type`
`hfc.protos.ledger.rwset.rwset_pb2.message_type`

```

hfc.protos.ledger.rwset.rwset_pb2.TxReadWriteSet
hfc.protos.ledger.rwset.rwset_pb2.NsReadWriteSet
hfc.protos.ledger.rwset.rwset_pb2.CollectionHashedReadWriteSet
hfc.protos.ledger.rwset.rwset_pb2.TxPvtReadWriteSet
hfc.protos.ledger.rwset.rwset_pb2.NsPvtReadWriteSet
hfc.protos.ledger.rwset.rwset_pb2.CollectionPvtReadWriteSet
hfc.protos.ledger.rwset.rwset_pb2.has_options = True
hfc.protos.ledger.rwset.rwset_pb2._options

```

```
hfc.protos.ledger.rwset.rwset_pb2_grpc
```

```
hfc.protos.msp
```

Submodules

```
hfc.protos.msp.identities_pb2
```

Module Contents

```

hfc.protos.msp.identities_pb2._b
hfc.protos.msp.identities_pb2._sym_db
hfc.protos.msp.identities_pb2.DESRIPTOR
hfc.protos.msp.identities_pb2._SERIALIZEDIDENTITY
hfc.protos.msp.identities_pb2._SERIALIZEDIDEMIXIDENTITY
hfc.protos.msp.identities_pb2.SerializedIdentity
hfc.protos.msp.identities_pb2.SerializedIdemixIdentity
hfc.protos.msp.identities_pb2._options

```

```
hfc.protos.msp.identities_pb2_grpc
```

```
hfc.protos.msp.msp_config_pb2
```

Module Contents

```

hfc.protos.msp.msp_config_pb2._b
hfc.protos.msp.msp_config_pb2._sym_db
hfc.protos.msp.msp_config_pb2.DESRIPTOR
hfc.protos.msp.msp_config_pb2._MSPCONFIG
hfc.protos.msp.msp_config_pb2._FABRICMSPCONFIG
hfc.protos.msp.msp_config_pb2._FABRICCRYPTOCONFIG

```

hfc.protos.msp.msp_config_pb2._IDEMIXMSPCONFIG
hfc.protos.msp.msp_config_pb2._IDEMIXMSPSIGNERCONFIG
hfc.protos.msp.msp_config_pb2._SIGNINGIDENTITYINFO
hfc.protos.msp.msp_config_pb2._KEYINFO
hfc.protos.msp.msp_config_pb2._FABRICOUIDENTIFIER
hfc.protos.msp.msp_config_pb2._FABRICNODEOUS
hfc.protos.msp.msp_config_pb2.message_type
hfc.protos.msp.msp_config_pb2.message_type
hfc.protos.msp.msp_config_pb2.message_type
hfc.protos.msp.msp_config_pb2.message_type
hfc.protos.msp.msp_config_pb2.message_type
hfc.protos.msp.msp_config_pb2.message_type
hfc.protos.msp.msp_config_pb2.message_type
hfc.protos.msp.msp_config_pb2.MSPConfig
hfc.protos.msp.msp_config_pb2.FabricMSPConfig
hfc.protos.msp.msp_config_pb2.FabricCryptoConfig
hfc.protos.msp.msp_config_pb2.IdemixMSPConfig
hfc.protos.msp.msp_config_pb2.IdemixMSPSignerConfig
hfc.protos.msp.msp_config_pb2.SigningIdentityInfo
hfc.protos.msp.msp_config_pb2.KeyInfo
hfc.protos.msp.msp_config_pb2.FabricOUIIdentifier
hfc.protos.msp.msp_config_pb2.FabricNodeOUs
hfc.protos.msp.msp_config_pb2._options

hfc.protos.msp.msp_config_pb2_grpc

hfc.protos.msp.msp_principal_pb2

Module Contents

hfc.protos.msp.msp_principal_pb2._b
hfc.protos.msp.msp_principal_pb2._sym_db
hfc.protos.msp.msp_principal_pb2.DESRIPTOR
hfc.protos.msp.msp_principal_pb2._MSPPRINCIPAL_CLASSIFICATION
hfc.protos.msp.msp_principal_pb2._MSPROLE_MSPROLETYPE
hfc.protos.msp.msp_principal_pb2._MSPIDENTITYANONYMITY_MSPIDENTITYANONYMITYTYPE
hfc.protos.msp.msp_principal_pb2._MSPPRINCIPAL

hfc.protos.msp.msp_principal_pb2._ORGANIZATIONUNIT
 hfc.protos.msp.msp_principal_pb2._MSPROLE
 hfc.protos.msp.msp_principal_pb2._MSPIDENTITYANONYMITY
 hfc.protos.msp.msp_principal_pb2._COMBINEDPRINCIPAL
 hfc.protos.msp.msp_principal_pb2.enum_type
 hfc.protos.msp.msp_principal_pb2.containing_type
 hfc.protos.msp.msp_principal_pb2.enum_type
 hfc.protos.msp.msp_principal_pb2.containing_type
 hfc.protos.msp.msp_principal_pb2.enum_type
 hfc.protos.msp.msp_principal_pb2.containing_type
 hfc.protos.msp.msp_principal_pb2.message_type
 hfc.protos.msp.msp_principal_pb2.MSPPrincipal
 hfc.protos.msp.msp_principal_pb2.OrganizationUnit
 hfc.protos.msp.msp_principal_pb2.MSPRole
 hfc.protos.msp.msp_principal_pb2.MSPIdentityAnonymity
 hfc.protos.msp.msp_principal_pb2.CombinedPrincipal
 hfc.protos.msp.msp_principal_pb2._options

hfc.protos.msp.msp_principal_pb2_grpc

hfc.protos.orderer

Submodules

hfc.protos.orderer.ab_pb2

Module Contents

hfc.protos.orderer.ab_pb2._b
 hfc.protos.orderer.ab_pb2._sym_db
 hfc.protos.orderer.ab_pb2.DESRIPTOR
 hfc.protos.orderer.ab_pb2._SEEKINFO_SEEKBEHAVIOR
 hfc.protos.orderer.ab_pb2._BROADCASTRESPONSE
 hfc.protos.orderer.ab_pb2._SEEKNEWEST
 hfc.protos.orderer.ab_pb2._SEEKOLDEST
 hfc.protos.orderer.ab_pb2._SEEKSPECIFIED
 hfc.protos.orderer.ab_pb2._SEEKPOSITION
 hfc.protos.orderer.ab_pb2._SEEKINFO
 hfc.protos.orderer.ab_pb2._DELIVERRESPONSE

hfc.protos.orderer.ab_pb2.enum_type
hfc.protos.orderer.ab_pb2.message_type
hfc.protos.orderer.ab_pb2.message_type
hfc.protos.orderer.ab_pb2.message_type
hfc.protos.orderer.ab_pb2.containing_oneof
hfc.protos.orderer.ab_pb2.containing_oneof
hfc.protos.orderer.ab_pb2.containing_oneof
hfc.protos.orderer.ab_pb2.message_type
hfc.protos.orderer.ab_pb2.message_type
hfc.protos.orderer.ab_pb2.enum_type
hfc.protos.orderer.ab_pb2.containing_type
hfc.protos.orderer.ab_pb2.enum_type
hfc.protos.orderer.ab_pb2.message_type
hfc.protos.orderer.ab_pb2.containing_oneof
hfc.protos.orderer.ab_pb2.containing_oneof
hfc.protos.orderer.ab_pb2.BroadcastResponse
hfc.protos.orderer.ab_pb2.SeekNewest
hfc.protos.orderer.ab_pb2.SeekOldest
hfc.protos.orderer.ab_pb2.SeekSpecified
hfc.protos.orderer.ab_pb2.SeekPosition
hfc.protos.orderer.ab_pb2.SeekInfo
hfc.protos.orderer.ab_pb2.DeliverResponse
hfc.protos.orderer.ab_pb2._options
hfc.protos.orderer.ab_pb2._ATOMICBROADCAST

hfc.protos.orderer.ab_pb2_grpc

Module Contents

Classes

AtomicBroadcastStub(channel)

AtomicBroadcastServicer()

Functions

```
add_AtomicBroadcastServicer_to_server(servicer,
server)
```

```
class hfc.protos.orderer.ab_pb2_grpc.AtomicBroadcastStub (channel)
    Bases: object
```

```
class hfc.protos.orderer.ab_pb2_grpc.AtomicBroadcastServicer
    Bases: object
```

```
    Broadcast (self, request_iterator, context)
```

broadcast receives a reply of Acknowledgement for each common.Envelope in order, indicating success or type of failure

```
    Deliver (self, request_iterator, context)
```

deliver first requires an Envelope of type DELIVER_SEEK_INFO with Payload data as a mashaed Seek-Info message, then a stream of block replies is received.

```
hfc.protos.orderer.ab_pb2_grpc.add_AtomicBroadcastServicer_to_server (servicer,
                                                                    server)
```

hfc.protos.orderer.cluster_pb2

Module Contents

```
hfc.protos.orderer.cluster_pb2._b
hfc.protos.orderer.cluster_pb2.__sym_db
hfc.protos.orderer.cluster_pb2.DESCRIPTOR
hfc.protos.orderer.cluster_pb2.__STEPREQUEST
hfc.protos.orderer.cluster_pb2.__STEPRESPONSE
hfc.protos.orderer.cluster_pb2.__SUBMITREQUEST
hfc.protos.orderer.cluster_pb2.__SUBMITRESPONSE
hfc.protos.orderer.cluster_pb2.message_type
hfc.protos.orderer.cluster_pb2.enum_type
hfc.protos.orderer.cluster_pb2.StepRequest
hfc.protos.orderer.cluster_pb2.StepResponse
hfc.protos.orderer.cluster_pb2.SubmitRequest
hfc.protos.orderer.cluster_pb2.SubmitResponse
hfc.protos.orderer.cluster_pb2.__options
hfc.protos.orderer.cluster_pb2.__CLUSTER
```

`hfc.protos.orderer.cluster_pb2_grpc`

Module Contents

Classes

<code>ClusterStub(channel)</code>	Cluster defines communication between cluster members.
<code>ClusterServicer()</code>	Cluster defines communication between cluster members.

Functions

<code>add_ClusterServicer_to_server(servicer, server)</code>
--

class `hfc.protos.orderer.cluster_pb2_grpc.ClusterStub(channel)`

Bases: object

Cluster defines communication between cluster members.

class `hfc.protos.orderer.cluster_pb2_grpc.ClusterServicer`

Bases: object

Cluster defines communication between cluster members.

Submit (*self, request_iterator, context*)

Submit submits transactions to a cluster member

Step (*self, request, context*)

Step passes an implementation-specific message to another cluster member.

`hfc.protos.orderer.cluster_pb2_grpc.add_ClusterServicer_to_server(servicer, server)`

`hfc.protos.orderer.configuration_pb2`

Module Contents

`hfc.protos.orderer.configuration_pb2._b`
`hfc.protos.orderer.configuration_pb2._sym_db`
`hfc.protos.orderer.configuration_pb2.DESRIPTOR`
`hfc.protos.orderer.configuration_pb2._CONSENSUSTYPE`
`hfc.protos.orderer.configuration_pb2._BATCHSIZE`
`hfc.protos.orderer.configuration_pb2._BATCHTIMEOUT`
`hfc.protos.orderer.configuration_pb2._KAFKABROKERS`
`hfc.protos.orderer.configuration_pb2._CHANNELRESTRICTIONS`
`hfc.protos.orderer.configuration_pb2.ConsensusType`
`hfc.protos.orderer.configuration_pb2.BatchSize`

[hfc.protos.orderer.configuration_pb2.BatchTimeout](#)
[hfc.protos.orderer.configuration_pb2.KafkaBrokers](#)
[hfc.protos.orderer.configuration_pb2.ChannelRestrictions](#)
[hfc.protos.orderer.configuration_pb2._options](#)

[hfc.protos.orderer.configuration_pb2_grpc](#)

[hfc.protos.orderer.kafka_pb2](#)

Module Contents

[hfc.protos.orderer.kafka_pb2._b](#)
[hfc.protos.orderer.kafka_pb2._sym_db](#)
[hfc.protos.orderer.kafka_pb2.DESRIPTOR](#)
[hfc.protos.orderer.kafka_pb2._KAFKAMESSAGEREGULAR_CLASS](#)
[hfc.protos.orderer.kafka_pb2._KAFKAMESSAGE](#)
[hfc.protos.orderer.kafka_pb2._KAFKAMESSAGEREGULAR](#)
[hfc.protos.orderer.kafka_pb2._KAFKAMESSAGETIMETOCUT](#)
[hfc.protos.orderer.kafka_pb2._KAFKAMESSAGECONNECT](#)
[hfc.protos.orderer.kafka_pb2._KAFKAMETADATA](#)
[hfc.protos.orderer.kafka_pb2.message_type](#)
[hfc.protos.orderer.kafka_pb2.message_type](#)
[hfc.protos.orderer.kafka_pb2.message_type](#)
[hfc.protos.orderer.kafka_pb2.containing_oneof](#)
[hfc.protos.orderer.kafka_pb2.containing_oneof](#)
[hfc.protos.orderer.kafka_pb2.containing_oneof](#)
[hfc.protos.orderer.kafka_pb2.enum_type](#)
[hfc.protos.orderer.kafka_pb2.containing_type](#)
[hfc.protos.orderer.kafka_pb2.KafkaMessage](#)
[hfc.protos.orderer.kafka_pb2.KafkaMessageRegular](#)
[hfc.protos.orderer.kafka_pb2.KafkaMessageTimeToCut](#)
[hfc.protos.orderer.kafka_pb2.KafkaMessageConnect](#)
[hfc.protos.orderer.kafka_pb2.KafkaMetadata](#)
[hfc.protos.orderer.kafka_pb2._options](#)

`hfc.protos.orderer.kafka_pb2_grpc`

`hfc.protos.peer`

Subpackages

`hfc.protos.peer.lifecycle`

Submodules

`hfc.protos.peer.lifecycle.lifecycle_pb2`

Module Contents

`hfc.protos.peer.lifecycle.lifecycle_pb2._b`

`hfc.protos.peer.lifecycle.lifecycle_pb2._sym_db`

`hfc.protos.peer.lifecycle.lifecycle_pb2.DESRIPTOR`

`hfc.protos.peer.lifecycle.lifecycle_pb2._INSTALLCHAINCODEARGS`

`hfc.protos.peer.lifecycle.lifecycle_pb2._INSTALLCHAINCODERESULT`

`hfc.protos.peer.lifecycle.lifecycle_pb2._QUERYINSTALLEDCHAINCODEARGS`

`hfc.protos.peer.lifecycle.lifecycle_pb2._QUERYINSTALLEDCHAINCODERESULT`

`hfc.protos.peer.lifecycle.lifecycle_pb2.InstallChaincodeArgs`

`hfc.protos.peer.lifecycle.lifecycle_pb2.InstallChaincodeResult`

`hfc.protos.peer.lifecycle.lifecycle_pb2.QueryInstalledChaincodeArgs`

`hfc.protos.peer.lifecycle.lifecycle_pb2.QueryInstalledChaincodeResult`

`hfc.protos.peer.lifecycle.lifecycle_pb2._options`

`hfc.protos.peer.lifecycle.lifecycle_pb2_grpc`

Submodules

`hfc.protos.peer.admin_pb2`

Module Contents

`hfc.protos.peer.admin_pb2._b`

`hfc.protos.peer.admin_pb2._sym_db`

`hfc.protos.peer.admin_pb2.DESRIPTOR`

`hfc.protos.peer.admin_pb2._SERVERSTATUS_STATUSCODE`

`hfc.protos.peer.admin_pb2._SERVERSTATUS`

`hfc.protos.peer.admin_pb2._LOGLEVELREQUEST`

```

hfc.protos.peer.admin_pb2._LOGLEVELRESPONSE
hfc.protos.peer.admin_pb2._LOGSPECREQUEST
hfc.protos.peer.admin_pb2._LOGSPECRESPONSE
hfc.protos.peer.admin_pb2._ADMINOPERATION
hfc.protos.peer.admin_pb2.enum_type
hfc.protos.peer.admin_pb2.containing_type
hfc.protos.peer.admin_pb2.message_type
hfc.protos.peer.admin_pb2.message_type
hfc.protos.peer.admin_pb2.containing_oneof
hfc.protos.peer.admin_pb2.containing_oneof
hfc.protos.peer.admin_pb2.ServerStatus
hfc.protos.peer.admin_pb2.LogLevelRequest
hfc.protos.peer.admin_pb2.LogLevelResponse
hfc.protos.peer.admin_pb2.LogSpecRequest
hfc.protos.peer.admin_pb2.LogSpecResponse
hfc.protos.peer.admin_pb2.AdminOperation
hfc.protos.peer.admin_pb2._options
hfc.protos.peer.admin_pb2._ADMIN

```

```
hfc.protos.peer.admin_pb2_grpc
```

Module Contents

Classes

<code>AdminStub(channel)</code>	Interface exported by the server.
<code>AdminServicer()</code>	Interface exported by the server.

Functions

<code>add_AdminServicer_to_server(servicer, server)</code>
--

```
class hfc.protos.peer.admin_pb2_grpc.AdminStub (channel)
```

Bases: object

Interface exported by the server.

```
class hfc.protos.peer.admin_pb2_grpc.AdminServicer
```

Bases: object

Interface exported by the server.

GetStatus (*self, request, context*)

StartServer (*self, request, context*)

GetModuleLogLevel (*self, request, context*)

SetModuleLogLevel (*self, request, context*)

RevertLogLevels (*self, request, context*)

GetLogSpec (*self, request, context*)

SetLogSpec (*self, request, context*)

`hfc.protos.peer.admin_pb2_grpc.add_AdminServicer_to_server` (*servicer, server*)

`hfc.protos.peer.chaincode_event_pb2`

Module Contents

`hfc.protos.peer.chaincode_event_pb2._b`

`hfc.protos.peer.chaincode_event_pb2._sym_db`

`hfc.protos.peer.chaincode_event_pb2.DESRIPTOR`

`hfc.protos.peer.chaincode_event_pb2._CHAINCODEEVENT`

`hfc.protos.peer.chaincode_event_pb2.ChaincodeEvent`

`hfc.protos.peer.chaincode_event_pb2._options`

`hfc.protos.peer.chaincode_event_pb2_grpc`

`hfc.protos.peer.chaincode_pb2`

Module Contents

`hfc.protos.peer.chaincode_pb2._b`

`hfc.protos.peer.chaincode_pb2._sym_db`

`hfc.protos.peer.chaincode_pb2.DESRIPTOR`

`hfc.protos.peer.chaincode_pb2._CONFIDENTIALITYLEVEL`

`hfc.protos.peer.chaincode_pb2.ConfidentialityLevel`

`hfc.protos.peer.chaincode_pb2.PUBLIC = 0`

`hfc.protos.peer.chaincode_pb2.CONFIDENTIAL = 1`

`hfc.protos.peer.chaincode_pb2._CHAINCODESPEC_TYPE`

`hfc.protos.peer.chaincode_pb2._CHAINCODEDEPLOYMENTSPEC_EXECUTIONENVIRONMENT`

`hfc.protos.peer.chaincode_pb2._CHAINCODEID`

`hfc.protos.peer.chaincode_pb2._CHAINCODEINPUT_DECORATIONSENTRY`

`hfc.protos.peer.chaincode_pb2._CHAINCODEINPUT`

`hfc.protos.peer.chaincode_pb2._CHAINCODESPEC`

`hfc.protos.peer.chaincode_pb2._CHAINCODEDEPLOYMENTSPEC`

hfc.protos.peer.chaincode_pb2._CHAINCODEINVOCATIONSPEC
 hfc.protos.peer.chaincode_pb2._LIFECYCLEEVENT
 hfc.protos.peer.chaincode_pb2._CSDATA
 hfc.protos.peer.chaincode_pb2._CHAINCODEDATA
 hfc.protos.peer.chaincode_pb2.containing_type
 hfc.protos.peer.chaincode_pb2.message_type
 hfc.protos.peer.chaincode_pb2.enum_type
 hfc.protos.peer.chaincode_pb2.message_type
 hfc.protos.peer.chaincode_pb2.message_type
 hfc.protos.peer.chaincode_pb2.containing_type
 hfc.protos.peer.chaincode_pb2.message_type
 hfc.protos.peer.chaincode_pb2.enum_type
 hfc.protos.peer.chaincode_pb2.containing_type
 hfc.protos.peer.chaincode_pb2.message_type
 hfc.protos.peer.chaincode_pb2.message_type
 hfc.protos.peer.chaincode_pb2.message_type
 hfc.protos.peer.chaincode_pb2.message_type
 hfc.protos.peer.chaincode_pb2.ChaincodeID
 hfc.protos.peer.chaincode_pb2.ChaincodeInput
 hfc.protos.peer.chaincode_pb2.ChaincodeSpec
 hfc.protos.peer.chaincode_pb2.ChaincodeDeploymentSpec
 hfc.protos.peer.chaincode_pb2.ChaincodeInvocationSpec
 hfc.protos.peer.chaincode_pb2.LifecycleEvent
 hfc.protos.peer.chaincode_pb2.CSDData
 hfc.protos.peer.chaincode_pb2.ChaincodeData
 hfc.protos.peer.chaincode_pb2._options
 hfc.protos.peer.chaincode_pb2._options

hfc.protos.peer.chaincode_pb2_grpc

hfc.protos.peer.chaincode_shim_pb2

Module Contents

hfc.protos.peer.chaincode_shim_pb2._b
 hfc.protos.peer.chaincode_shim_pb2._sym_db
 hfc.protos.peer.chaincode_shim_pb2.DESRIPTOR
 hfc.protos.peer.chaincode_shim_pb2._CHAINCODEMESSAGE_TYPE

hfc.protos.peer.chaincode_shim_pb2.**_CHAINCODEMESSAGE**
hfc.protos.peer.chaincode_shim_pb2.**_GETSTATE**
hfc.protos.peer.chaincode_shim_pb2.**_GETSTATEMETADATA**
hfc.protos.peer.chaincode_shim_pb2.**_PUTSTATE**
hfc.protos.peer.chaincode_shim_pb2.**_PUTSTATEMETADATA**
hfc.protos.peer.chaincode_shim_pb2.**_DELSTATE**
hfc.protos.peer.chaincode_shim_pb2.**_GETSTATEBYRANGE**
hfc.protos.peer.chaincode_shim_pb2.**_GETQUERYRESULT**
hfc.protos.peer.chaincode_shim_pb2.**_QUERYMETADATA**
hfc.protos.peer.chaincode_shim_pb2.**_GETHISTORYFORKEY**
hfc.protos.peer.chaincode_shim_pb2.**_QUERYSTATENEXT**
hfc.protos.peer.chaincode_shim_pb2.**_QUERYSTATECLOSE**
hfc.protos.peer.chaincode_shim_pb2.**_QUERYRESULTBYTES**
hfc.protos.peer.chaincode_shim_pb2.**_QUERYRESPONSE**
hfc.protos.peer.chaincode_shim_pb2.**_QUERYRESPONSEMETADATA**
hfc.protos.peer.chaincode_shim_pb2.**_STATEMETADATA**
hfc.protos.peer.chaincode_shim_pb2.**_STATEMETADATARESULT**
hfc.protos.peer.chaincode_shim_pb2.**enum_type**
hfc.protos.peer.chaincode_shim_pb2.**message_type**
hfc.protos.peer.chaincode_shim_pb2.**message_type**
hfc.protos.peer.chaincode_shim_pb2.**message_type**
hfc.protos.peer.chaincode_shim_pb2.**containing_type**
hfc.protos.peer.chaincode_shim_pb2.**message_type**
hfc.protos.peer.chaincode_shim_pb2.**message_type**
hfc.protos.peer.chaincode_shim_pb2.**message_type**
hfc.protos.peer.chaincode_shim_pb2.**ChaincodeMessage**
hfc.protos.peer.chaincode_shim_pb2.**GetState**
hfc.protos.peer.chaincode_shim_pb2.**GetStateMetadata**
hfc.protos.peer.chaincode_shim_pb2.**PutState**
hfc.protos.peer.chaincode_shim_pb2.**PutStateMetadata**
hfc.protos.peer.chaincode_shim_pb2.**DelState**
hfc.protos.peer.chaincode_shim_pb2.**GetStateByRange**
hfc.protos.peer.chaincode_shim_pb2.**GetQueryResult**
hfc.protos.peer.chaincode_shim_pb2.**QueryMetadata**
hfc.protos.peer.chaincode_shim_pb2.**GetHistoryForKey**
hfc.protos.peer.chaincode_shim_pb2.**QueryStateNext**

`hfc.protos.peer.chaincode_shim_pb2.QueryStateClose`
`hfc.protos.peer.chaincode_shim_pb2.QueryResultBytes`
`hfc.protos.peer.chaincode_shim_pb2.QueryResponse`
`hfc.protos.peer.chaincode_shim_pb2.QueryResponseMetadata`
`hfc.protos.peer.chaincode_shim_pb2.StateMetadata`
`hfc.protos.peer.chaincode_shim_pb2.StateMetadataResult`
`hfc.protos.peer.chaincode_shim_pb2._options`
`hfc.protos.peer.chaincode_shim_pb2._CHAINCODESUPPORT`

`hfc.protos.peer.chaincode_shim_pb2_grpc`

Module Contents

Classes

<code>ChaincodeSupportStub(channel)</code>	Interface that provides support to chaincode execution. ChaincodeContext
<code>ChaincodeSupportServicer()</code>	Interface that provides support to chaincode execution. ChaincodeContext

Functions

`add_ChaincodeSupportServicer_to_server(servicer, server)`

class `hfc.protos.peer.chaincode_shim_pb2_grpc.ChaincodeSupportStub(channel)`
 Bases: object

Interface that provides support to chaincode execution. ChaincodeContext provides the context necessary for the server to respond appropriately.

class `hfc.protos.peer.chaincode_shim_pb2_grpc.ChaincodeSupportServicer`
 Bases: object

Interface that provides support to chaincode execution. ChaincodeContext provides the context necessary for the server to respond appropriately.

Register (*self, request_iterator, context*)

`hfc.protos.peer.chaincode_shim_pb2_grpc.add_ChaincodeSupportServicer_to_server(servicer, server)`

hfc.protos.peer.configuration_pb2

Module Contents

hfc.protos.peer.configuration_pb2._b
hfc.protos.peer.configuration_pb2._sym_db
hfc.protos.peer.configuration_pb2.DESRIPTOR
hfc.protos.peer.configuration_pb2._ANCHORPEERS
hfc.protos.peer.configuration_pb2._ANCHORPEER
hfc.protos.peer.configuration_pb2._APIRESOURCE
hfc.protos.peer.configuration_pb2._ACLS_ACLSENTRY
hfc.protos.peer.configuration_pb2._ACLS
hfc.protos.peer.configuration_pb2.message_type
hfc.protos.peer.configuration_pb2.message_type
hfc.protos.peer.configuration_pb2.containing_type
hfc.protos.peer.configuration_pb2.message_type
hfc.protos.peer.configuration_pb2.AnchorPeers
hfc.protos.peer.configuration_pb2.AnchorPeer
hfc.protos.peer.configuration_pb2.APIResource
hfc.protos.peer.configuration_pb2.ACLs
hfc.protos.peer.configuration_pb2._options
hfc.protos.peer.configuration_pb2._options

hfc.protos.peer.configuration_pb2_grpc

hfc.protos.peer.events_pb2

Module Contents

hfc.protos.peer.events_pb2._b
hfc.protos.peer.events_pb2._sym_db
hfc.protos.peer.events_pb2.DESRIPTOR
hfc.protos.peer.events_pb2._FILTEREDBLOCK
hfc.protos.peer.events_pb2._FILTEREDTRANSACTION
hfc.protos.peer.events_pb2._FILTEREDTRANSACTIONACTIONS
hfc.protos.peer.events_pb2._FILTEREDCHAINCODEACTION
hfc.protos.peer.events_pb2._DELIVERRESPONSE
hfc.protos.peer.events_pb2.message_type
hfc.protos.peer.events_pb2.enum_type

```

hfc.protos.peer.events_pb2.enum_type
hfc.protos.peer.events_pb2.message_type
hfc.protos.peer.events_pb2.containing_oneof
hfc.protos.peer.events_pb2.message_type
hfc.protos.peer.events_pb2.message_type
hfc.protos.peer.events_pb2.enum_type
hfc.protos.peer.events_pb2.message_type
hfc.protos.peer.events_pb2.message_type
hfc.protos.peer.events_pb2.containing_oneof
hfc.protos.peer.events_pb2.containing_oneof
hfc.protos.peer.events_pb2.containing_oneof
hfc.protos.peer.events_pb2.FilteredBlock
hfc.protos.peer.events_pb2.FilteredTransaction
hfc.protos.peer.events_pb2.FilteredTransactionActions
hfc.protos.peer.events_pb2.FilteredChaincodeAction
hfc.protos.peer.events_pb2.DeliverResponse
hfc.protos.peer.events_pb2._options
hfc.protos.peer.events_pb2._DELIVER

```

```
hfc.protos.peer.events_pb2_grpc
```

Module Contents

Classes

```
DeliverStub(channel)
```

```
DeliverServicer()
```

Functions

```
add_DeliverServicer_to_server(servicer,
server)
```

```
class hfc.protos.peer.events_pb2_grpc.DeliverStub (channel)
    Bases: object
```

```
class hfc.protos.peer.events_pb2_grpc.DeliverServicer
    Bases: object
```

```
    Deliver (self, request_iterator, context)
```

```
        deliver first requires an Envelope of type ab.DELIVER_SEEK_INFO with Payload data as a marshaled
        orderer.SeekInfo message, then a stream of block replies is received
```

DeliverFiltered (*self, request_iterator, context*)

deliver first requires an Envelope of type `ab.DELIVER_SEEK_INFO` with Payload data as a marshaled `orderer.SeekInfo` message, then a stream of **filtered** block replies is received

`hfc.protos.peer.events_pb2_grpc.add_DeliverServicer_to_server` (*servicer, server*)

`hfc.protos.peer.peer_pb2`

Module Contents

`hfc.protos.peer.peer_pb2._b`
`hfc.protos.peer.peer_pb2._sym_db`
`hfc.protos.peer.peer_pb2.DESRIPTOR`
`hfc.protos.peer.peer_pb2._PEERID`
`hfc.protos.peer.peer_pb2._PEERENDPOINT`
`hfc.protos.peer.peer_pb2.message_type`
`hfc.protos.peer.peer_pb2.PeerID`
`hfc.protos.peer.peer_pb2.PeerEndpoint`
`hfc.protos.peer.peer_pb2._options`
`hfc.protos.peer.peer_pb2._ENDORSER`

`hfc.protos.peer.peer_pb2_grpc`

Module Contents

Classes

`EndorserStub`(*channel*)

`EndorserServicer`()

Functions

`add_EndorserServicer_to_server`(*servicer, server*)

class `hfc.protos.peer.peer_pb2_grpc.EndorserStub` (*channel*)
Bases: `object`

class `hfc.protos.peer.peer_pb2_grpc.EndorserServicer`
Bases: `object`

ProcessProposal (*self, request, context*)

`hfc.protos.peer.peer_pb2_grpc.add_EndorserServicer_to_server` (*servicer, server*)

hfc.protos.peer.policy_pb2**Module Contents**

hfc.protos.peer.policy_pb2._b
 hfc.protos.peer.policy_pb2._sym_db
 hfc.protos.peer.policy_pb2.DESRIPTOR
 hfc.protos.peer.policy_pb2._APPLICATIONPOLICY
 hfc.protos.peer.policy_pb2.message_type
 hfc.protos.peer.policy_pb2.containing_oneof
 hfc.protos.peer.policy_pb2.containing_oneof
 hfc.protos.peer.policy_pb2.ApplicationPolicy
 hfc.protos.peer.policy_pb2._options

hfc.protos.peer.policy_pb2_grpc**hfc.protos.peer.proposal_pb2****Module Contents**

hfc.protos.peer.proposal_pb2._b
 hfc.protos.peer.proposal_pb2._sym_db
 hfc.protos.peer.proposal_pb2.DESRIPTOR
 hfc.protos.peer.proposal_pb2._SIGNEDPROPOSAL
 hfc.protos.peer.proposal_pb2._PROPOSAL
 hfc.protos.peer.proposal_pb2._CHAINCODEHEADEREXTENSION
 hfc.protos.peer.proposal_pb2._CHAINCODEPROPOSALPAYLOAD_TRANSIENTMAPENTRY
 hfc.protos.peer.proposal_pb2._CHAINCODEPROPOSALPAYLOAD
 hfc.protos.peer.proposal_pb2._CHAINCODEACTION
 hfc.protos.peer.proposal_pb2.message_type
 hfc.protos.peer.proposal_pb2.containing_type
 hfc.protos.peer.proposal_pb2.message_type
 hfc.protos.peer.proposal_pb2.message_type
 hfc.protos.peer.proposal_pb2.message_type
 hfc.protos.peer.proposal_pb2.message_type
 hfc.protos.peer.proposal_pb2.SignedProposal
 hfc.protos.peer.proposal_pb2.Proposal
 hfc.protos.peer.proposal_pb2.ChaincodeHeaderExtension
 hfc.protos.peer.proposal_pb2.ChaincodeProposalPayload

hfc.protos.peer.proposal_pb2.ChaincodeAction
hfc.protos.peer.proposal_pb2._options
hfc.protos.peer.proposal_pb2._options

hfc.protos.peer.proposal_pb2_grpc

hfc.protos.peer.proposal_response_pb2

Module Contents

hfc.protos.peer.proposal_response_pb2._b
hfc.protos.peer.proposal_response_pb2._sym_db
hfc.protos.peer.proposal_response_pb2.DESRIPTOR
hfc.protos.peer.proposal_response_pb2._PROPOSALRESPONSE
hfc.protos.peer.proposal_response_pb2._RESPONSE
hfc.protos.peer.proposal_response_pb2._PROPOSALRESPONSEPAYLOAD
hfc.protos.peer.proposal_response_pb2._ENDORSEMENT
hfc.protos.peer.proposal_response_pb2.message_type
hfc.protos.peer.proposal_response_pb2.message_type
hfc.protos.peer.proposal_response_pb2.message_type
hfc.protos.peer.proposal_response_pb2.ProposalResponse
hfc.protos.peer.proposal_response_pb2.Response
hfc.protos.peer.proposal_response_pb2.ProposalResponsePayload
hfc.protos.peer.proposal_response_pb2.Endorsement
hfc.protos.peer.proposal_response_pb2._options

hfc.protos.peer.proposal_response_pb2_grpc

hfc.protos.peer.query_pb2

Module Contents

hfc.protos.peer.query_pb2._b
hfc.protos.peer.query_pb2._sym_db
hfc.protos.peer.query_pb2.DESRIPTOR
hfc.protos.peer.query_pb2._CHAINCODEQUERYRESPONSE
hfc.protos.peer.query_pb2._CHAINCODEINFO
hfc.protos.peer.query_pb2._CHANNELQUERYRESPONSE
hfc.protos.peer.query_pb2._CHANNELINFO

`hfc.protos.peer.query_pb2.message_type`
`hfc.protos.peer.query_pb2.message_type`
`hfc.protos.peer.query_pb2.ChaincodeQueryResponse`
`hfc.protos.peer.query_pb2.ChaincodeInfo`
`hfc.protos.peer.query_pb2.ChannelQueryResponse`
`hfc.protos.peer.query_pb2.ChannelInfo`
`hfc.protos.peer.query_pb2._options`

`hfc.protos.peer.query_pb2_grpc`

`hfc.protos.peer.resources_pb2`

Module Contents

`hfc.protos.peer.resources_pb2._b`
`hfc.protos.peer.resources_pb2._sym_db`
`hfc.protos.peer.resources_pb2.DESRIPTOR`
`hfc.protos.peer.resources_pb2._CHAINCODEIDENTIFIER`
`hfc.protos.peer.resources_pb2._CHAINCODEVALIDATION`
`hfc.protos.peer.resources_pb2._VSCCARGS`
`hfc.protos.peer.resources_pb2._CHAINCODEENDORSEMENT`
`hfc.protos.peer.resources_pb2._CONFIGTREE`
`hfc.protos.peer.resources_pb2.message_type`
`hfc.protos.peer.resources_pb2.message_type`
`hfc.protos.peer.resources_pb2.ChaincodeIdentifier`
`hfc.protos.peer.resources_pb2.ChaincodeValidation`
`hfc.protos.peer.resources_pb2.VSCCArgs`
`hfc.protos.peer.resources_pb2.ChaincodeEndorsement`
`hfc.protos.peer.resources_pb2.ConfigTree`
`hfc.protos.peer.resources_pb2._options`

`hfc.protos.peer.resources_pb2_grpc`

`hfc.protos.peer.signed_cc_dep_spec_pb2`

Module Contents

`hfc.protos.peer.signed_cc_dep_spec_pb2._b`
`hfc.protos.peer.signed_cc_dep_spec_pb2._sym_db`

hfc.protos.peer.signed_cc_dep_spec_pb2.DESRIPTOR
hfc.protos.peer.signed_cc_dep_spec_pb2._SIGNEDCHAINCODEDEPLOYMENTSPEC
hfc.protos.peer.signed_cc_dep_spec_pb2.message_type
hfc.protos.peer.signed_cc_dep_spec_pb2.SignedChaincodeDeploymentSpec
hfc.protos.peer.signed_cc_dep_spec_pb2._options

hfc.protos.peer.signed_cc_dep_spec_pb2_grpc

hfc.protos.peer.transaction_pb2

Module Contents

hfc.protos.peer.transaction_pb2._b
hfc.protos.peer.transaction_pb2._sym_db
hfc.protos.peer.transaction_pb2.DESRIPTOR
hfc.protos.peer.transaction_pb2._TXVALIDATIONCODE
hfc.protos.peer.transaction_pb2.TxValidationCode
hfc.protos.peer.transaction_pb2._METADATAKEYS
hfc.protos.peer.transaction_pb2.MetaDataKeys
hfc.protos.peer.transaction_pb2.VALID = 0
hfc.protos.peer.transaction_pb2.NIL_ENVELOPE = 1
hfc.protos.peer.transaction_pb2.BAD_PAYLOAD = 2
hfc.protos.peer.transaction_pb2.BAD_COMMON_HEADER = 3
hfc.protos.peer.transaction_pb2.BAD_CREATOR_SIGNATURE = 4
hfc.protos.peer.transaction_pb2.INVALID_ENDORSER_TRANSACTION = 5
hfc.protos.peer.transaction_pb2.INVALID_CONFIG_TRANSACTION = 6
hfc.protos.peer.transaction_pb2.UNSUPPORTED_TX_PAYLOAD = 7
hfc.protos.peer.transaction_pb2.BAD_PROPOSAL_TXID = 8
hfc.protos.peer.transaction_pb2.DUPLICATE_TXID = 9
hfc.protos.peer.transaction_pb2.ENDORSEMENT_POLICY_FAILURE = 10
hfc.protos.peer.transaction_pb2.MVCC_READ_CONFLICT = 11
hfc.protos.peer.transaction_pb2.PHANTOM_READ_CONFLICT = 12
hfc.protos.peer.transaction_pb2.UNKNOWN_TX_TYPE = 13
hfc.protos.peer.transaction_pb2.TARGET_CHAIN_NOT_FOUND = 14
hfc.protos.peer.transaction_pb2.MARSHAL_TX_ERROR = 15
hfc.protos.peer.transaction_pb2.NIL_TXACTION = 16
hfc.protos.peer.transaction_pb2.EXPIRED_CHAINCODE = 17
hfc.protos.peer.transaction_pb2.CHAINCODE_VERSION_CONFLICT = 18


```

hfc.protos.peer.transaction_pb2.BAD_HEADER_EXTENSION = 19
hfc.protos.peer.transaction_pb2.BAD_CHANNEL_HEADER = 20
hfc.protos.peer.transaction_pb2.BAD_RESPONSE_PAYLOAD = 21
hfc.protos.peer.transaction_pb2.BAD_RWSET = 22
hfc.protos.peer.transaction_pb2.ILLEGAL_WRITESET = 23
hfc.protos.peer.transaction_pb2.INVALID_WRITESET = 24
hfc.protos.peer.transaction_pb2.NOT_VALIDATED = 254
hfc.protos.peer.transaction_pb2.INVALID_OTHER_REASON = 255
hfc.protos.peer.transaction_pb2.VALIDATION_PARAMETER = 0
hfc.protos.peer.transaction_pb2._SIGNEDTRANSACTION
hfc.protos.peer.transaction_pb2._PROCESSEDTRANSACTION
hfc.protos.peer.transaction_pb2._TRANSACTION
hfc.protos.peer.transaction_pb2._TRANSACTIONACTION
hfc.protos.peer.transaction_pb2._CHAINCODEACTIONPAYLOAD
hfc.protos.peer.transaction_pb2._CHAINCODEENDORSEDACTION
hfc.protos.peer.transaction_pb2.message_type
hfc.protos.peer.transaction_pb2.message_type
hfc.protos.peer.transaction_pb2.message_type
hfc.protos.peer.transaction_pb2.message_type
hfc.protos.peer.transaction_pb2.SignedTransaction
hfc.protos.peer.transaction_pb2.ProcessedTransaction
hfc.protos.peer.transaction_pb2.Transaction
hfc.protos.peer.transaction_pb2.TransactionAction
hfc.protos.peer.transaction_pb2.ChaincodeActionPayload
hfc.protos.peer.transaction_pb2.ChaincodeEndorsedAction
hfc.protos.peer.transaction_pb2._options

```

```
hfc.protos.peer.transaction_pb2_grpc
```

```
hfc.protos.token
```

Submodules

```
hfc.protos.token.expectations_pb2
```

Module Contents

```
hfc.protos.token.expectations_pb2._b
```

```
hfc.protos.token.expectations_pb2._sym_db
```

hfc.protos.token.expectations_pb2.DESRIPTOR
hfc.protos.token.expectations_pb2._TOKENEXPECTATION
hfc.protos.token.expectations_pb2._PLAINEXPECTATION
hfc.protos.token.expectations_pb2._PLAINTOKENEXPECTATION
hfc.protos.token.expectations_pb2.message_type
hfc.protos.token.expectations_pb2.containing_oneof
hfc.protos.token.expectations_pb2.message_type
hfc.protos.token.expectations_pb2.message_type
hfc.protos.token.expectations_pb2.containing_oneof
hfc.protos.token.expectations_pb2.containing_oneof
hfc.protos.token.expectations_pb2.message_type
hfc.protos.token.expectations_pb2.TokenExpectation
hfc.protos.token.expectations_pb2.PlainExpectation
hfc.protos.token.expectations_pb2.PlainTokenExpectation
hfc.protos.token.expectations_pb2._options

hfc.protos.token.expectations_pb2_grpc

hfc.protos.token.prover_pb2

Module Contents

hfc.protos.token.prover_pb2._b
hfc.protos.token.prover_pb2._sym_db
hfc.protos.token.prover_pb2.DESRIPTOR
hfc.protos.token.prover_pb2._TOKENTOISSUE
hfc.protos.token.prover_pb2._RECIPIENTTRANSFERSHARE
hfc.protos.token.prover_pb2._TOKENOUTPUT
hfc.protos.token.prover_pb2._UNSPENTTOKENS
hfc.protos.token.prover_pb2._LISTREQUEST
hfc.protos.token.prover_pb2._IMPORTREQUEST
hfc.protos.token.prover_pb2._TRANSFERREQUEST
hfc.protos.token.prover_pb2._REDEEMREQUEST
hfc.protos.token.prover_pb2._ALLOWANCERECIPIENTSHARE
hfc.protos.token.prover_pb2._APPROVEREQUEST
hfc.protos.token.prover_pb2._EXPECTATIONREQUEST
hfc.protos.token.prover_pb2._HEADER
hfc.protos.token.prover_pb2._COMMAND

hfc.protos.token.prover_pb2.**containing_oneof**
hfc.protos.token.prover_pb2.**containing_oneof**
hfc.protos.token.prover_pb2.**containing_oneof**
hfc.protos.token.prover_pb2.**TokenToIssue**
hfc.protos.token.prover_pb2.**RecipientTransferShare**
hfc.protos.token.prover_pb2.**TokenOutput**
hfc.protos.token.prover_pb2.**UnspentTokens**
hfc.protos.token.prover_pb2.**ListRequest**
hfc.protos.token.prover_pb2.**ImportRequest**
hfc.protos.token.prover_pb2.**TransferRequest**
hfc.protos.token.prover_pb2.**RedeemRequest**
hfc.protos.token.prover_pb2.**AllowanceRecipientShare**
hfc.protos.token.prover_pb2.**ApproveRequest**
hfc.protos.token.prover_pb2.**ExpectationRequest**
hfc.protos.token.prover_pb2.**Header**
hfc.protos.token.prover_pb2.**Command**
hfc.protos.token.prover_pb2.**SignedCommand**
hfc.protos.token.prover_pb2.**CommandResponseHeader**
hfc.protos.token.prover_pb2.**Error**
hfc.protos.token.prover_pb2.**CommandResponse**
hfc.protos.token.prover_pb2.**SignedCommandResponse**
hfc.protos.token.prover_pb2.**_options**
hfc.protos.token.prover_pb2.**_PROVER**

hfc.protos.token.prover_pb2_grpc

Module Contents

Classes

<i>ProverStub</i> (channel)	Prover provides support to clients for the creation of FabToken transactions,
<i>ProverServicer</i> ()	Prover provides support to clients for the creation of FabToken transactions,

Functions

```
add_ProverServicer_to_server(servicer,
server)
```

```
class hfc.protos.token.prover_pb2_grpc.ProverStub (channel)
    Bases: object
```

Prover provides support to clients for the creation of FabToken transactions, and to query the ledger.

```
class hfc.protos.token.prover_pb2_grpc.ProverServicer
    Bases: object
```

Prover provides support to clients for the creation of FabToken transactions, and to query the ledger.

```
ProcessCommand (self, request, context)
```

ProcessCommand processes the passed command ensuring proper access control. The returned response allows the client to understand if the operation was successfully executed and if not, the response reports the reason of the failure.

```
hfc.protos.token.prover_pb2_grpc.add_ProverServicer_to_server (servicer, server)
```

```
hfc.protos.token.transaction_pb2
```

Module Contents

```
hfc.protos.token.transaction_pb2._b
hfc.protos.token.transaction_pb2._sym_db
hfc.protos.token.transaction_pb2.DESRIPTOR
hfc.protos.token.transaction_pb2._TOKENTRANSACTION
hfc.protos.token.transaction_pb2._PLAINTOKENACTION
hfc.protos.token.transaction_pb2._PLAINIMPORT
hfc.protos.token.transaction_pb2._PLAINTRANSFER
hfc.protos.token.transaction_pb2._PLAINAPPROVE
hfc.protos.token.transaction_pb2._PLAINTRANSFERFROM
hfc.protos.token.transaction_pb2._PLAINOUTPUT
hfc.protos.token.transaction_pb2._INPUTID
hfc.protos.token.transaction_pb2._PLAINDELEGATEDOUTPUT
hfc.protos.token.transaction_pb2.message_type
hfc.protos.token.transaction_pb2.containing_oneof
hfc.protos.token.transaction_pb2.message_type
hfc.protos.token.transaction_pb2.message_type
hfc.protos.token.transaction_pb2.message_type
hfc.protos.token.transaction_pb2.message_type
hfc.protos.token.transaction_pb2.message_type
```

hfc.protos.token.transaction_pb2.containing_oneof
hfc.protos.token.transaction_pb2.containing_oneof
hfc.protos.token.transaction_pb2.containing_oneof
hfc.protos.token.transaction_pb2.containing_oneof
hfc.protos.token.transaction_pb2.containing_oneof
hfc.protos.token.transaction_pb2.message_type
hfc.protos.token.transaction_pb2.message_type
hfc.protos.token.transaction_pb2.message_type
hfc.protos.token.transaction_pb2.message_type
hfc.protos.token.transaction_pb2.message_type
hfc.protos.token.transaction_pb2.message_type
hfc.protos.token.transaction_pb2.message_type
hfc.protos.token.transaction_pb2.message_type
hfc.protos.token.transaction_pb2.message_type
hfc.protos.token.transaction_pb2.TokenTransaction
hfc.protos.token.transaction_pb2.PlainTokenAction
hfc.protos.token.transaction_pb2.PlainImport
hfc.protos.token.transaction_pb2.PlainTransfer
hfc.protos.token.transaction_pb2.PlainApprove
hfc.protos.token.transaction_pb2.PlainTransferFrom
hfc.protos.token.transaction_pb2.PlainOutput
hfc.protos.token.transaction_pb2.InputId
hfc.protos.token.transaction_pb2.PlainDelegatedOutput
hfc.protos.token.transaction_pb2._options

hfc.protos.token.transaction_pb2_grpc

hfc.protos.transientstore

Submodules

hfc.protos.transientstore.transientstore_pb2

Module Contents

hfc.protos.transientstore.transientstore_pb2._b
hfc.protos.transientstore.transientstore_pb2._sym_db
hfc.protos.transientstore.transientstore_pb2.DESRIPTOR
hfc.protos.transientstore.transientstore_pb2._TXPVTREADWRITESETWITHCONFIGINFO_COLLECTIONCON

```

hfc.protos.transientstore.transientstore_pb2._TXPVTREADWRITESETWITHCONFIGINFO
hfc.protos.transientstore.transientstore_pb2.message_type
hfc.protos.transientstore.transientstore_pb2.containing_type
hfc.protos.transientstore.transientstore_pb2.message_type
hfc.protos.transientstore.transientstore_pb2.message_type
hfc.protos.transientstore.transientstore_pb2.TxPvtReadWriteSetWithConfigInfo
hfc.protos.transientstore.transientstore_pb2._options
hfc.protos.transientstore.transientstore_pb2._options

```

```
hfc.protos.transientstore.transientstore_pb2_grpc
```

Submodules

```
hfc.protos.utils
```

Module Contents

Functions

```
create_seek_info(start=None, stop=None, behavior='BLOCK_UNTIL_READY')
```

```
create_seek_payload(seek_header, seek_info)
```

```
create_cc_spec(chaincode_input, chaincode_id, type)
```

```
create_tx_payload(endorsements, tran_req)
```

```
create_envelope(seek_payload_sig, seek_payload_bytes)
```

```
hfc.protos.utils.create_seek_info (start=None, stop=None, behavior='BLOCK_UNTIL_READY')
```

```
hfc.protos.utils.create_seek_payload (seek_header, seek_info)
```

```
hfc.protos.utils.create_cc_spec (chaincode_input, chaincode_id, type)
```

```
hfc.protos.utils.create_tx_payload (endorsements, tran_req)
```

```
hfc.protos.utils.create_envelope (seek_payload_sig, seek_payload_bytes)
```

`hfc.util`

Subpackages

`hfc.util.crypto`

Submodules

`hfc.util.crypto.crypto`

Module Contents

Classes

<code>Key()</code>	An abstract base class for Key.
<code>AsymmetricKey()</code>	An asymmetric key.
<code>Crypto()</code>	An abstract base class for crypto.
<code>Ecies(security_level=CURVE_P_256_Size, hash_algorithm=SHA2)</code>	A crypto implementation based on ECDSA and SHA.

Functions

<code>generate_nonce(size)</code>	Generate a secure random for cryptographic use.
<code>ecies(security_level=CURVE_P_256_Size, hash_algorithm=SHA2)</code>	Factory method for creating a Ecies instance.

`hfc.util.crypto.crypto.DEFAULT_NONCE_SIZE = 24`

`hfc.util.crypto.crypto.CURVE_P_256_Size = 256`

`hfc.util.crypto.crypto.CURVE_P_384_Size = 384`

`hfc.util.crypto.crypto.SHA2 = SHA2`

`hfc.util.crypto.crypto.SHA3 = SHA3`

`hfc.util.crypto.crypto.AES_KEY_LENGTH = 32`

`hfc.util.crypto.crypto.HMAC_KEY_LENGTH = 32`

`hfc.util.crypto.crypto.IV_LENGTH = 16`

class `hfc.util.crypto.crypto.Key`

Bases: `object`

An abstract base class for Key.

Key represents a base cryptographic key. It can be symmetric or asymmetric. In asymmetric case, the private key can retrieve public key with the corresponding method.

A key can be referenced via the Subject Key Identifier (SKI) with DER or PEM encoding.

abstract `is_symmetric` (*self*)

Return if this key is with symmetric crypt, i.e. whether it's a symmetric key.

Returns True or False

abstract get_SKI (*self*)
Return the SKI string

Returns string represent the SKI

class `hfc.util.crypto.crypto.AsymmetricKey`

Bases: `hfc.util.crypto.crypto.Key`

An asymmetric key.

Can be a public key or private key, the private key can retrieve public key with the corresponding method.

abstract is_private (*self*)
Return if this key is private key

Returns True or False

abstract get_public_key (*self*)
Get the corresponding public key for this private key.

If this key is already a public one, then return itself.

Returns Public key

class `hfc.util.crypto.crypto.Crypto`

Bases: `object`

An abstract base class for crypto.

abstract generate_private_key (*self*)
Generate asymmetric key pair.

Returns An private key object which include public key object.

abstract encrypt (*self*, *public_key*, *message*)
Encrypt the message by encryption public key.

Parameters

- **public_key** – Encryption public key
- **message** – message need encrypt

Returns An object including secure context

abstract decrypt (*self*, *private_key*, *cipher_text*)
Decrypt the cipher text by encryption private key.

Parameters

- **private_key** – Encryption private key
- **cipher_text** – Cipher text received

Returns An object including secure context

abstract sign (*self*, *private_key*, *message*)
Sign the origin message by signing private key.

Parameters

- **private_key** – Signing private key
- **message** – Origin message

Returns An object including secure context

abstract verify (*self, public_key, message, signature*)
Verify the signature by signing public key.

Parameters

- **public_key** – Signing public key
- **message** – Origin message
- **signature** – Signature of message

Returns A boolean True as valid

static generate_nonce (*size*)
Generate a secure random for cryptographic use.

Parameters **size** – Number of bytes for the nonce

Returns Generated random bytes

`hfc.util.crypto.crypto.generate_nonce` (*size*)
Generate a secure random for cryptographic use.

Parameters **size** – Number of bytes for the nonce

Returns Generated random bytes

class `hfc.util.crypto.crypto.Ecies` (*security_level=CURVE_P_256_Size*,
hash_algorithm=SHA2)

Bases: `hfc.util.crypto.crypto.Crypto`

A crypto implementation based on ECDSA and SHA.

property hash (*self*)
Get hash function

Returns hash function

sign (*self, private_key, message*)
ECDSA sign message.

Parameters

- **private_key** – private key
- **message** – message to sign

Returns signature

verify (*self, public_key, message, signature*)
ECDSA verify signature.

Parameters

- **public_key** – Signing public key
- **message** – Origin message
- **signature** – Signature of message

Returns verify result boolean, True means valid

_prevent_malleability (*self, sig*)

_check_malleability (*self, sig*)

generate_private_key (*self*)
ECDSA key pair generation by current curve.

Returns A private key object which include public key object.

decrypt (*self, private_key, cipher_text*)
ECIES decrypt cipher text.

First restore the ephemeral public key from bytes(97 bytes for 384, 65 bytes for 256).

Then derived a shared key based ecdh, using the key based hkdf to generate aes key and hmac key, using hmac-sha3 to verify the hmac bytes. Last using aes-256-cfb to decrypt the bytes.

Parameters

- **private_key** – private key
- **cipher_text** – cipher text

Returns plain text

encrypt (*self, public_key, plain_text*)
ECIES encrypt plain text.

First create a ephemeral ecdsa key pair, then serialize the public key for part of result. Then derived a shared key based ecdh, using the key based hkdf to generate aes key and hmac key, using aes-256-cfb to generate the part of result. Last using hmac-sha3 and the part of previous step to generate last part of result.

Parameters

- **public_key** – public key
- **plain_text** – plain text

Returns cipher text

generate_csr (*self, private_key, subject_name, extensions=None*)
Generate certificate signing request.

Parameters

- **private_key** – Private key
- **subject_name** (*x509.Name*) – Subject name
- **extensions** – (Default value = None)

return: x509.CertificateSigningRequest

`hfc.util.crypto.crypto.ecies` (*security_level=CURVE_P_256_Size, hash_algorithm=SHA2*)
Factory method for creating a Ecies instance.

Parameters

- **security_level** – Security level (Default value = CURVE_P_256_Size)
- **hash_algorithm** – Hash algorithm

Returns A Ecies instance (Default value = SHA2)

Package Contents

Classes

<code>NullHandler(level=NOTSET)</code>	Handler instances dispatch logging events to specific destinations.
--	---

class `hfc.util.crypto.NullHandler` (*level=NOTSET*)

Bases: `logging.Handler`

Handler instances dispatch logging events to specific destinations.

The base handler class. Acts as a placeholder which defines the `Handler` interface. Handlers can optionally use `Formatter` instances to format records as desired. By default, no formatter is specified; in this case, the ‘raw’ message as determined by `record.message` is logged.

emit (*self, record*)

Do whatever it takes to actually log the specified logging record.

This version is intended to be implemented by subclasses and so raises a `NotImplementedError`.

Submodules

`hfc.util.channel`

Module Contents

Functions

<code>create_grpc_channel(target, cert_file=None, client_key=None, client_cert=None, opts=None)</code>	Construct a grpc channel.
--	---------------------------

`hfc.util.channel.create_grpc_channel` (*target, cert_file=None, client_key=None, client_cert=None, opts=None*)

Construct a grpc channel.

Parameters

- **target** – server address include host:port
- **cert_file** – ssl/tls root cert file for the connection (Default value = None)
- **opts** – grpc channel options `grpc.default_authority:` default authority
`grpc.ssl_target_name_override:` ssl target name override
- **client_key** – client key (Default value = None)
- **client_cert** – client certificate (Default value = None)

Returns `grpc channel`

`hfc.util.keyvaluestore`

Module Contents

Classes

<code>KeyValueStore()</code>	An key value store for blockchain application state persistence.
<code>FileKeyValueStore(path)</code>	A key value store implementation based file system.

Functions

<code>__make_dir(path)</code>	
<code>file_key_value_store(path)</code>	Factory method for creating file key value store.

class `hfc.util.keyvaluestore.KeyValueStore`Bases: `object`

An key value store for blockchain application state persistence.

abstract `set_value(self, key, value)`

Set a value with a specific key.

Parameters

- **key** – key
- **value** – value

abstract `get_value(self, key)`

Get a value with a specific key.

Parameters **key** – key**Returns** value**abstract** `async_set_value(self, key, value, scheduler=None)`

Set a value with a specific key.

Parameters

- **scheduler** – scheduler
- **key** – key
- **value** – value

:return:a future object

abstract `async_get_value(self, key, scheduler=None)`

Get a value with a specific key.

Parameters

- **scheduler** – scheduler
- **key** – key

:return:a future object

class `hfc.util.keyvaluestore.FileKeyValueStore` (*path*)

Bases: `hfc.util.keyvaluestore.KeyValueStore`

A key value store implementation based file system.

set_value (*self, key, value*)

Set a value with a specific key.

Args: key: key value: value

Returns: True when success Raises: File manipulate exceptions

get_value (*self, key*)

Get a value with a specific key.

Parameters **key** – key

Returns value

async_get_value (*self, key, scheduler=None*)

Get a value with a specific key.

Parameters

- **scheduler** – scheduler
- **key** – key

:return:a future object

async_set_value (*self, key, value, scheduler=None*)

Set a value with a specific key.

Parameters

- **scheduler** – scheduler
- **key** – key
- **value** – value

:return:a future object

get_attrs (*self*)

__str__ (*self*)

Return str(self).

`hfc.util.keyvaluestore._make_dir` (*path*)

`hfc.util.keyvaluestore.file_key_value_store` (*path*)

Factory method for creating file key value store.

Parameters **path** – path

:return an instance of file key value store

hfc.util.policies**Module Contents****Classes**

String2Dict()

Dict2String()

Functions

s2d()

```
hfc.util.policies.s2d_grammar = ?value: e | role | DIGIT -> number
```

```
dot: "." dash: "-" name: /[wd-$$&+,:;=?@#|<>^*()%!]+/ mspid: WORD role: "" name dot mspid
"" or: "OR" and: "AND" outof: "OutOf" logic: or | and | outof e : logic "(" [value ("," value)* "]"
```

```
%import common.WORD %import common.LETTER %import common.DIGIT %import com-
mon.WS %ignore WS
```

```
class hfc.util.policies.String2Dict
```

```
    Bases: lark.Transformer
```

```
    unique_list_of_dict (self, list_of_dict)
```

```
    get_logic (self, args, n)
```

```
    get_outof (self, items)
```

```
    name (self, items)
```

```
    role (self, items)
```

```
    logic (self, items)
```

```
    dot (self, *args)
```

```
    dash (self, *args)
```

```
    mspid (self, items)
```

```
    number (self, items)
```

```
    e (self, items)
```

```
class hfc.util.policies.Dict2String
```

```
    Bases: object
```

```
    roles = []
```

```
    get_policy (self, policy)
```

```
    parse (self, policy)
```

```
hfc.util.policies.s2d()
```

```
hfc.util.policies.d2s
```

hfc.util.utils

Module Contents

Classes

`zeroTimeContextManager()`

Functions

<code>proto_str(x)</code>	
<code>create_serialized_identity(user)</code>	Create serialized identity from user.
<code>build_header(creator, channel_header, nonce)</code>	This function will build the common header.
<code>build_channel_header(type, tx_id, channel_id, timestamp, epoch=0, extension=None, tls_cert_hash=None)</code>	Build channel header.
<code>string_to_signature(string_signatures)</code>	Check if signatures are already in protobuf format.
<code>current_timestamp()</code>	Get current timestamp.
<code>extract_channel_config(configtx_proto_envelope)</code>	Extracts the protobuf 'ConfigUpdate' object out of the 'ConfigEnvelope'.
<code>build_cc_proposal(cci_spec, header, transient_map)</code>	Create an chaincode transaction proposal
<code>sign_proposal(tx_context, proposal)</code>	Sign a proposal
<code>send_transaction_proposal(proposal, tx_context, peers)</code>	Send transaction proposal
<code>send_transaction(orderers, tran_req, tx_context)</code>	Send a transaction to the chain's orderer service (one or more)
<code>sign_tran_payload(tx_context, tran_payload_bytes)</code>	Sign a transaction payload
<code>build_tx_req(ProposalResponses)</code>	Check the endorsements from peers
<code>send_install_proposal(tx_context, peers)</code>	Send install chaincode proposal
<code>zeroTarInfo(tarinfo)</code>	
<code>package_chaincode(cc_path, cc_type=CC_TYPE_GOLANG)</code>	Package all chaincode env into a tar.gz file
<code>pem_to_der(pem)</code>	
<code>stream_envelope(envelope)</code>	

`hfc.util.utils.CC_INSTALL = install``hfc.util.utils.CC_TYPE_GOLANG = GOLANG``hfc.util.utils._logger``hfc.util.utils.proto_str(x)``hfc.util.utils.proto_b``hfc.util.utils.create_serialized_identity(user)`

Create serialized identity from user.

Parameters `user` (*user object*) – The user object that should be serialized.**Returns** Protobuf SerializedIdentity of the given user object.

Return type `serialized_identity`

`hfc.util.utils.build_header(creator, channel_header, nonce)`

This function will build the common header.

Parameters

- **creator** (*protobuf SerializedIdentity*) – Serialized identity of the creator.
- **channel_header** (*protobuf ChannelHeader*) – ChannelHeader
- **nonce** (*str*) – Nonce that has been used for the tx_id.

Returns Returns created protobuf common header.

Return type `header`

`hfc.util.utils.build_channel_header(type, tx_id, channel_id, timestamp, epoch=0, extension=None, tls_cert_hash=None)`

Build channel header.

Parameters

- **type** (*common_pb2.HeaderType*) – type
- **tx_id** (*str*) – transaction id
- **channel_id** (*str*) – channel id
- **timestamp** (*grpc.timestamp*) – timestamp
- **epoch** (*int*) – epoch (Default value = 0)
- **extension** – extension (Default value = None)
- **tls_cert_hash** – (Default value = None)

Returns `common_proto.Header` instance

`hfc.util.utils.string_to_signature(string_signatures)`

Check if signatures are already in protobuf format.

Parameters **string_signatures** (*list*) – An list of protobuf ConfigSignatures either represented as or serialized as byte strings.

Returns List of protobuf ConfigSignatures.

Return type `list`

`hfc.util.utils.current_timestamp()`

Get current timestamp.

Returns Current timestamp.

`hfc.util.utils.extract_channel_config(configtx_proto_envelope)`

Extracts the protobuf 'ConfigUpdate' object out of the 'ConfigEnvelope'.

Parameters **configtx_proto_envelope** (*common_pb2.Envelope*) – The encoded bytes of the ConfigEnvelope protobuf.

Returns (*config_update*) The encoded bytes of the ConfigUpdate protobuf, ready to be signed

Return type `configtx_pb2.ConfigUpdateEnvelope.config_update`

Raises **ValueError** – If there is an error in protobuf_decode due to a wrong or not valid protobuf file a ValueError is raised.

`hfc.util.utils.build_cc_proposal(cci_spec, header, transient_map)`

Create an chaincode transaction proposal

Parameters

- **transient_map** – transient data map
- **cci_spec** – The spec
- **header** – header of the proposal

Returns The created proposal

`hfc.util.utils.sign_proposal(tx_context, proposal)`
Sign a proposal

Parameters

- **tx_context** – transaction context
- **proposal** – proposal to sign on

Returns Signed proposal

`hfc.util.utils.send_transaction_proposal(proposal, tx_context, peers)`
Send transaction proposal

Parameters

- **header** – header
- **tx_context** – transaction context
- **proposal** – transaction proposal
- **peers** – peers

Returns a list containing all the proposal response

`hfc.util.utils.send_transaction(orderers, tran_req, tx_context)`

Send a transaction to the chain's orderer service (one or more orderer endpoints) for consensus and committing to the ledger.

This call is asynchronous and the successful transaction commit is notified via a BLOCK or CHAINCODE event. This method must provide a mechanism for applications to attach event listeners to handle 'transaction submitted', 'transaction complete' and 'error' events.

Parameters

- **tx_context** – transaction context
- **orderers** – orderers
- **tran_req** (*TransactionRequest*) – The transaction object

Returns (EventEmitter) an handle to allow the application to attach event handlers on 'submitted', 'complete', and 'error'.

Return type EventEmitter

`hfc.util.utils.sign_tran_payload(tx_context, tran_payload_bytes)`
Sign a transaction payload

Parameters

- **signing_identity** – id to sign with
- **tran_payload** – transaction payload to sign on
- **tx_context** –
- **tran_payload_bytes** –

Returns Envelope

`hfc.util.utils.build_tx_req(ProposalResponses)`
Check the endorsements from peers

Parameters

- **reponses** – ProposalResponse from endorsers
- **ProposalResponses** –

Returns an instance of TXRequest

`hfc.util.utils.send_install_proposal(tx_context, peers)`
Send install chaincode proposal

Parameters

- **tx_context** – transaction context
- **peers** – peers to install chaincode

Returns a set of proposal response

`hfc.util.utils.zeroTarInfo(tarinfo)`

class `hfc.util.utils.zeroTimeContextManager`

Bases: object

`__enter__(self)`

`__exit__(self, type, value, traceback)`

`hfc.util.utils.package_chaincode(cc_path, cc_type=CC_TYPE_GOLANG)`
Package all chaincode env into a tar.gz file

Parameters

- **cc_path** – path to the chaincode
- **cc_type** – chaincode type (Default value = CC_TYPE_GOLANG)

Returns The chaincode pkg path or None

`hfc.util.utils.pem_to_der(pem)`

async `hfc.util.utils.stream_envelope(envelope)`

Package Contents

Classes

`NullHandler(level=NOTSET)`

Handler instances dispatch logging events to specific destinations.

class `hfc.util.NullHandler(level=NOTSET)`

Bases: `logging.Handler`

Handler instances dispatch logging events to specific destinations.

The base handler class. Acts as a placeholder which defines the Handler interface. Handlers can optionally use Formatter instances to format records as desired. By default, no formatter is specified; in this case, the 'raw' message as determined by `record.message` is logged.

emit (*self, record*)

Do whatever it takes to actually log the specified logging record.

This version is intended to be implemented by subclasses and so raises a `NotImplementedError`.

Submodules

hfc.version

Module Contents

`hfc.version.VERSION = 0.9.0`

Package Contents

`hfc.VERSION = 0.9.0`

PYTHON MODULE INDEX

h

hfc, 36
hfc.fabric, 36
hfc.fabric.block_decoder, 52
hfc.fabric.certificateAuthority, 60
hfc.fabric.channel, 36
hfc.fabric.channel.channel, 36
hfc.fabric.channel.channel_configuration,
42
hfc.fabric.channel.channel_eventhub, 42
hfc.fabric.channel.instantiation, 44
hfc.fabric.channel.invocation, 45
hfc.fabric.client, 61
hfc.fabric.config, 48
hfc.fabric.config.default, 48
hfc.fabric.msp, 48
hfc.fabric.orderer, 71
hfc.fabric.organization, 72
hfc.fabric.peer, 73
hfc.fabric.transaction, 48
hfc.fabric.transaction.tx_context, 48
hfc.fabric.transaction.tx_proposal_request,
49
hfc.fabric.user, 75
hfc.fabric_ca, 86
hfc.fabric_ca.affiliationService, 86
hfc.fabric_ca.caservice, 88
hfc.fabric_ca.certificateService, 93
hfc.fabric_ca.identityService, 94
hfc.fabric_network, 96
hfc.fabric_network.contract, 96
hfc.fabric_network.couchdbwalletstore,
97
hfc.fabric_network.gateway, 98
hfc.fabric_network.inmemorywalletstore,
99
hfc.fabric_network.network, 100
hfc.fabric_network.wallet, 100
hfc.protos, 101
hfc.protos.common, 101
hfc.protos.common.collection_pb2, 101
hfc.protos.common.collection_pb2_grpc,
102
hfc.protos.common.common_pb2, 102
hfc.protos.common.common_pb2_grpc, 104
hfc.protos.common.configtx_pb2, 104
hfc.protos.common.configtx_pb2_grpc, 106
hfc.protos.common.configuration_pb2, 106
hfc.protos.common.configuration_pb2_grpc,
107
hfc.protos.common.ledger_pb2, 107
hfc.protos.common.ledger_pb2_grpc, 107
hfc.protos.common.policies_pb2, 107
hfc.protos.common.policies_pb2_grpc, 108
hfc.protos.discovery, 108
hfc.protos.discovery.protocol_pb2, 108
hfc.protos.discovery.protocol_pb2_grpc,
111
hfc.protos.gossip, 112
hfc.protos.gossip.message_pb2, 112
hfc.protos.gossip.message_pb2_grpc, 116
hfc.protos.idemix, 117
hfc.protos.idemix.idemix_pb2, 117
hfc.protos.idemix.idemix_pb2_grpc, 118
hfc.protos.ledger, 118
hfc.protos.ledger.queryresult, 118
hfc.protos.ledger.queryresult.kv_query_result_pb2,
118
hfc.protos.ledger.queryresult.kv_query_result_pb2_grpc,
119
hfc.protos.ledger.rwset, 119
hfc.protos.ledger.rwset.kvrwset, 119
hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2,
119
hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2_grpc,
120
hfc.protos.ledger.rwset.rwset_pb2, 120
hfc.protos.ledger.rwset.rwset_pb2_grpc,
121
hfc.protos.msp, 121
hfc.protos.msp.identities_pb2, 121
hfc.protos.msp.identities_pb2_grpc, 121
hfc.protos.msp.msp_config_pb2, 121
hfc.protos.msp.msp_config_pb2_grpc, 122

hfc.protos.msp.msp_principal_pb2, 122
hfc.protos.msp.msp_principal_pb2_grpc, 123
hfc.protos.orderer, 123
hfc.protos.orderer.ab_pb2, 123
hfc.protos.orderer.ab_pb2_grpc, 124
hfc.protos.orderer.cluster_pb2, 125
hfc.protos.orderer.cluster_pb2_grpc, 126
hfc.protos.orderer.configuration_pb2, 126
hfc.protos.orderer.configuration_pb2_grpc, 127
hfc.protos.orderer.kafka_pb2, 127
hfc.protos.orderer.kafka_pb2_grpc, 128
hfc.protos.peer, 128
hfc.protos.peer.admin_pb2, 128
hfc.protos.peer.admin_pb2_grpc, 129
hfc.protos.peer.chaincode_event_pb2, 130
hfc.protos.peer.chaincode_event_pb2_grpc, 130
hfc.protos.peer.chaincode_pb2, 130
hfc.protos.peer.chaincode_pb2_grpc, 131
hfc.protos.peer.chaincode_shim_pb2, 131
hfc.protos.peer.chaincode_shim_pb2_grpc, 133
hfc.protos.peer.configuration_pb2, 134
hfc.protos.peer.configuration_pb2_grpc, 134
hfc.protos.peer.events_pb2, 134
hfc.protos.peer.events_pb2_grpc, 135
hfc.protos.peer.lifecycle, 128
hfc.protos.peer.lifecycle.lifecycle_pb2, 128
hfc.protos.peer.lifecycle.lifecycle_pb2_grpc, 128
hfc.protos.peer.peer_pb2, 136
hfc.protos.peer.peer_pb2_grpc, 136
hfc.protos.peer.policy_pb2, 137
hfc.protos.peer.policy_pb2_grpc, 137
hfc.protos.peer.proposal_pb2, 137
hfc.protos.peer.proposal_pb2_grpc, 138
hfc.protos.peer.proposal_response_pb2, 138
hfc.protos.peer.proposal_response_pb2_grpc, 138
hfc.protos.peer.query_pb2, 138
hfc.protos.peer.query_pb2_grpc, 139
hfc.protos.peer.resources_pb2, 139
hfc.protos.peer.resources_pb2_grpc, 139
hfc.protos.peer.signed_cc_dep_spec_pb2, 139
hfc.protos.peer.signed_cc_dep_spec_pb2_grpc, 140
hfc.protos.peer.transaction_pb2, 140
hfc.protos.peer.transaction_pb2_grpc, 141
hfc.protos.token, 141
hfc.protos.token.expectations_pb2, 141
hfc.protos.token.expectations_pb2_grpc, 142
hfc.protos.token.prover_pb2, 142
hfc.protos.token.prover_pb2_grpc, 144
hfc.protos.token.transaction_pb2, 145
hfc.protos.token.transaction_pb2_grpc, 146
hfc.protos.transientstore, 146
hfc.protos.transientstore.transientstore_pb2, 146
hfc.protos.transientstore.transientstore_pb2_grpc, 147
hfc.protos.utils, 147
hfc.util, 148
hfc.util.channel, 152
hfc.util.crypto, 148
hfc.util.crypto.crypto, 148
hfc.util.keyvaluestore, 153
hfc.util.policies, 155
hfc.util.utils, 156
hfc.version, 160

Symbols

- `_ACKNOWLEDGEMENT` (in module *hfc.protos.gossip.message_pb2*), 113
- `_ACLS` (in module *hfc.protos.peer.configuration_pb2*), 134
- `_ACLS_ACLSENTRY` (in module *hfc.protos.peer.configuration_pb2*), 134
- `_ADMIN` (in module *hfc.protos.peer.admin_pb2*), 129
- `_ADMINOPERATION` (in module *hfc.protos.peer.admin_pb2*), 129
- `_ALIVEMESSAGE` (in module *hfc.protos.gossip.message_pb2*), 112
- `_ALLOWANCERECIPIENTSHARE` (in module *hfc.protos.token.prover_pb2*), 142
- `_ANCHORPEER` (in module *hfc.protos.peer.configuration_pb2*), 134
- `_ANCHORPEERS` (in module *hfc.protos.peer.configuration_pb2*), 134
- `_APIRESOURCE` (in module *hfc.protos.peer.configuration_pb2*), 134
- `_APPLICATIONPOLICY` (in module *hfc.protos.peer.policy_pb2*), 137
- `_APPROVEREQUEST` (in module *hfc.protos.token.prover_pb2*), 142
- `_ATOMICBROADCAST` (in module *hfc.protos.orderer.ab_pb2*), 124
- `_AUTHINFO` (in module *hfc.protos.discovery.protocol_pb2*), 108
- `_BATCHSIZE` (in module *hfc.protos.orderer.configuration_pb2*), 126
- `_BATCHTIMEOUT` (in module *hfc.protos.orderer.configuration_pb2*), 126
- `_BLOCK` (in module *hfc.protos.common.common_pb2*), 103
- `_BLOCKCHAININFO` (in module *hfc.protos.common.ledger_pb2*), 107
- `_BLOCKDATA` (in module *hfc.protos.common.common_pb2*), 103
- `_BLOCKDATAHASHINGSTRUCTURE` (in module *hfc.protos.common.configuration_pb2*), 106
- `_BLOCKHEADER` (in module *hfc.protos.common.common_pb2*), 103
- `_BLOCKMETADATA` (in module *hfc.protos.common.common_pb2*), 103
- `_BLOCKMETADATAINDEX` (in module *hfc.protos.common.common_pb2*), 102
- `_BROADCASTRESPONSE` (in module *hfc.protos.orderer.ab_pb2*), 123
- `_CAPABILITIES` (in module *hfc.protos.common.configuration_pb2*), 106
- `_CAPABILITIES_CAPABILITIESENTRY` (in module *hfc.protos.common.configuration_pb2*), 106
- `_CAPABILITY` (in module *hfc.protos.common.configuration_pb2*), 106
- `_CDSDATA` (in module *hfc.protos.peer.chaincode_pb2*), 131
- `_CHAINCODE` (in module *hfc.protos.gossip.message_pb2*), 113
- `_CHAINCODEACTION` (in module *hfc.protos.peer.proposal_pb2*), 137
- `_CHAINCODEACTIONPAYLOAD` (in module *hfc.protos.peer.transaction_pb2*), 141
- `_CHAINCODECALL` (in module *hfc.protos.discovery.protocol_pb2*), 108
- `_CHAINCODEDATA` (in module *hfc.protos.peer.chaincode_pb2*), 131
- `_CHAINCODEDEPLOYMENTSPEC` (in module *hfc.protos.peer.chaincode_pb2*), 130
- `_CHAINCODEDEPLOYMENTSPEC_EXECUTIONENVIRONMENT` (in module *hfc.protos.peer.chaincode_pb2*), 130
- `_CHAINCODEENDORSEDACTION` (in module *hfc.protos.peer.transaction_pb2*), 141
- `_CHAINCODEENDORSEMENT` (in module *hfc.protos.peer.resources_pb2*), 139
- `_CHAINCODEEVENT` (in module *hfc.protos.peer.chaincode_event_pb2*), 130
- `_CHAINCODEHEADEREXTENSION` (in module *hfc.protos.peer.proposal_pb2*), 137
- `_CHAINCODEID` (in module *hfc.protos.peer.chaincode_pb2*), 130
- `_CHAINCODEIDENTIFIER` (in module *hfc.protos.peer.resources_pb2*), 139
- `_CHAINCODEINFO` (in module *hfc.protos.common.common_pb2*), 103

hfc.protos.peer.query_pb2), 138
 _CHAINCODEINPUT (in module *hfc.protos.peer.chaincode_pb2*), 130
 _CHAINCODEINPUT_DECORATIONSENTRY (in module *hfc.protos.peer.chaincode_pb2*), 130
 _CHAINCODEINTEREST (in module *hfc.protos.discovery.protocol_pb2*), 108
 _CHAINCODEINVOCATIONSPEC (in module *hfc.protos.peer.chaincode_pb2*), 130
 _CHAINCODEMESSAGE (in module *hfc.protos.peer.chaincode_shim_pb2*), 131
 _CHAINCODEMESSAGE_TYPE (in module *hfc.protos.peer.chaincode_shim_pb2*), 131
 _CHAINCODEPROPOSALPAYLOAD (in module *hfc.protos.peer.proposal_pb2*), 137
 _CHAINCODEPROPOSALPAYLOAD_TRANSIENTMAPENTRY (in module *hfc.protos.peer.proposal_pb2*), 137
 _CHAINCODEQUERY (in module *hfc.protos.discovery.protocol_pb2*), 108
 _CHAINCODEQUERYRESPONSE (in module *hfc.protos.peer.query_pb2*), 138
 _CHAINCODEQUERYRESULT (in module *hfc.protos.discovery.protocol_pb2*), 108
 _CHAINCODESPEC (in module *hfc.protos.peer.chaincode_pb2*), 130
 _CHAINCODESPEC_TYPE (in module *hfc.protos.peer.chaincode_pb2*), 130
 _CHAINCODESUPPORT (in module *hfc.protos.peer.chaincode_shim_pb2*), 133
 _CHAINCODEVALIDATION (in module *hfc.protos.peer.resources_pb2*), 139
 _CHANNELHEADER (in module *hfc.protos.common.common_pb2*), 103
 _CHANNELINFO (in module *hfc.protos.peer.query_pb2*), 138
 _CHANNELQUERYRESPONSE (in module *hfc.protos.peer.query_pb2*), 138
 _CHANNELRESTRICTIONS (in module *hfc.protos.orderer.configuration_pb2*), 126
 _CLUSTER (in module *hfc.protos.orderer.cluster_pb2*), 125
 _COLLECTIONCONFIG (in module *hfc.protos.common.collection_pb2*), 102
 _COLLECTIONCONFIGPACKAGE (in module *hfc.protos.common.collection_pb2*), 101
 _COLLECTIONCRITERIA (in module *hfc.protos.common.collection_pb2*), 102
 _COLLECTIONHASHEDREADWRITESSET (in module *hfc.protos.ledger.rwset.rwset_pb2*), 120
 _COLLECTIONPOLICYCONFIG (in module *hfc.protos.common.collection_pb2*), 102
 _COLLECTIONPVTREADWRITESSET (in module *hfc.protos.ledger.rwset.rwset_pb2*), 120
 _COMBINEDPRINCIPAL (in module *hfc.protos.msp.msp_principal_pb2*), 123
 _COMMAND (in module *hfc.protos.token.prover_pb2*), 142
 _COMMANDRESPONSE (in module *hfc.protos.token.prover_pb2*), 143
 _COMMANDRESPONSEHEADER (in module *hfc.protos.token.prover_pb2*), 143
 _CONFIDENTIALITYLEVEL (in module *hfc.protos.peer.chaincode_pb2*), 130
 _CONFIG (in module *hfc.protos.common.configtx_pb2*), 104
 _CONFIGENVELOPE (in module *hfc.protos.common.configtx_pb2*), 104
 _CONFIGGROUP (in module *hfc.protos.common.configtx_pb2*), 104
 _CONFIGGROUPSCHEMA (in module *hfc.protos.common.configtx_pb2*), 104
 _CONFIGGROUPSCHEMA_GROUPENTRY (in module *hfc.protos.common.configtx_pb2*), 104
 _CONFIGGROUPSCHEMA_POLICIESENTRY (in module *hfc.protos.common.configtx_pb2*), 104
 _CONFIGGROUPSCHEMA_VALUEENTRY (in module *hfc.protos.common.configtx_pb2*), 104
 _CONFIGGROUP_GROUPENTRY (in module *hfc.protos.common.configtx_pb2*), 104
 _CONFIGGROUP_POLICIESENTRY (in module *hfc.protos.common.configtx_pb2*), 104
 _CONFIGGROUP_VALUEENTRY (in module *hfc.protos.common.configtx_pb2*), 104
 _CONFIGPOLICY (in module *hfc.protos.common.configtx_pb2*), 104
 _CONFIGPOLICYSCHEMA (in module *hfc.protos.common.configtx_pb2*), 104
 _CONFIGQUERY (in module *hfc.protos.discovery.protocol_pb2*), 108
 _CONFIGRESULT (in module *hfc.protos.discovery.protocol_pb2*), 108
 _CONFIGRESULT_MSPENTRY (in module *hfc.protos.discovery.protocol_pb2*), 108
 _CONFIGRESULT_ORDERERSENTRY (in module *hfc.protos.discovery.protocol_pb2*), 108
 _CONFIGSIGNATURE (in module *hfc.protos.common.configtx_pb2*), 105
 _CONFIGTREE (in module *hfc.protos.peer.resources_pb2*), 139
 _CONFIGUPDATE (in module *hfc.protos.common.configtx_pb2*), 104
 _CONFIGUPDATEENVELOPE (in module *hfc.protos.common.configtx_pb2*), 104
 _CONFIGUPDATE_ISOLATEDDATAENTRY (in module *hfc.protos.common.configtx_pb2*), 104
 _CONFIGVALUE (in module *hfc.protos.common.configtx_pb2*), 104
 _CONFIGVALUESHEMA (in module *hfc.protos.common.configtx_pb2*), 104

<code>__INPUTID</code>	(in module <i>hfc.protos.token.transaction_pb2</i>), 145	<code>__LIFECYCLEEVENT</code>	(in module <i>hfc.protos.peer.chaincode_pb2</i>), 131
<code>__INSTALLCHAINCODEARGS</code>	(in module <i>hfc.protos.peer.lifecycle.lifecycle_pb2</i>), 128	<code>__LISTREQUEST</code>	(in module <i>hfc.protos.token.prover_pb2</i>), 142
<code>__INSTALLCHAINCODERESULT</code>	(in module <i>hfc.protos.peer.lifecycle.lifecycle_pb2</i>), 128	<code>__LOCALPEERQUERY</code>	(in module <i>hfc.protos.discovery.protocol_pb2</i>), 109
<code>__ISSUERKEY</code>	(in module <i>hfc.protos.idemix.idemix_pb2</i>), 117	<code>__LOGLEVELREQUEST</code>	(in module <i>hfc.protos.peer.admin_pb2</i>), 128
<code>__ISSUERPUBLICKEY</code>	(in module <i>hfc.protos.idemix.idemix_pb2</i>), 117	<code>__LOGLEVELRESPONSE</code>	(in module <i>hfc.protos.peer.admin_pb2</i>), 128
<code>__KAFKABROKERS</code>	(in module <i>hfc.protos.orderer.configuration_pb2</i>), 126	<code>__LOGSPECREQUEST</code>	(in module <i>hfc.protos.peer.admin_pb2</i>), 129
<code>__KAFKAMESSAGE</code>	(in module <i>hfc.protos.orderer.kafka_pb2</i>), 127	<code>__LOGSPECRESPONSE</code>	(in module <i>hfc.protos.peer.admin_pb2</i>), 129
<code>__KAFKAMESSAGECONNECT</code>	(in module <i>hfc.protos.orderer.kafka_pb2</i>), 127	<code>__MEMBER</code>	(in module <i>hfc.protos.gossip.message_pb2</i>), 113
<code>__KAFKAMESSAGEREGULAR</code>	(in module <i>hfc.protos.orderer.kafka_pb2</i>), 127	<code>__MEMBERSHIPREQUEST</code>	(in module <i>hfc.protos.gossip.message_pb2</i>), 112
<code>__KAFKAMESSAGEREGULAR_CLASS</code>	(in module <i>hfc.protos.orderer.kafka_pb2</i>), 127	<code>__MEMBERSHIPRESPONSE</code>	(in module <i>hfc.protos.gossip.message_pb2</i>), 113
<code>__KAFKAMESSAGETIMETOCUT</code>	(in module <i>hfc.protos.orderer.kafka_pb2</i>), 127	<code>__METADATA</code>	(in module <i>hfc.protos.common.common_pb2</i>), 103
<code>__KAFKAMETADATA</code>	(in module <i>hfc.protos.orderer.kafka_pb2</i>), 127	<code>__METADATAKEYS</code>	(in module <i>hfc.protos.peer.transaction_pb2</i>), 140
<code>__KEYINFO</code>	(in module <i>hfc.protos.msp.msp_config_pb2</i>), 122	<code>__METADATASIGNATURE</code>	(in module <i>hfc.protos.common.common_pb2</i>), 103
<code>__KEYMODIFICATION</code>	(in module <i>hfc.protos.ledger.queryresult.kv_query_result_pb2</i>), 118	<code>__MSPCONFIG</code>	(in module <i>hfc.protos.msp.msp_config_pb2</i>), 121
<code>__KV</code>	(in module <i>hfc.protos.ledger.queryresult.kv_query_result_pb2</i>), 118	<code>__MSPIDENTITYANONYMITY</code>	(in module <i>hfc.protos.msp.msp_principal_pb2</i>), 123
<code>__KVREAD</code>	(in module <i>hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2</i>), 119	<code>__MSPIDENTITYANONYMITY_MSPIDENTITYANONYMITYTYPE</code>	(in module <i>hfc.protos.msp.msp_principal_pb2</i>), 122
<code>__KVREADHASH</code>	(in module <i>hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2</i>), 119	<code>__MSPPRINCIPAL</code>	(in module <i>hfc.protos.msp.msp_principal_pb2</i>), 122
<code>__KVRWSET</code>	(in module <i>hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2</i>), 119	<code>__MSPPRINCIPAL_CLASSIFICATION</code>	(in module <i>hfc.protos.msp.msp_principal_pb2</i>), 122
<code>__KVWRITE</code>	(in module <i>hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2</i>), 119	<code>__MSPROLE</code>	(in module <i>hfc.protos.msp.msp_principal_pb2</i>), 123
<code>__KVWRITEHASH</code>	(in module <i>hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2</i>), 119	<code>__MSPROLE_MSPROLETYPE</code>	(in module <i>hfc.protos.msp.msp_principal_pb2</i>), 122
<code>__LASTCONFIG</code>	(in module <i>hfc.protos.common.common_pb2</i>), 103	<code>__NONREVOCATIONPROOF</code>	(in module <i>hfc.protos.idemix.idemix_pb2</i>), 117
<code>__LAYOUT</code>	(in module <i>hfc.protos.discovery.protocol_pb2</i>), 109	<code>__NSPVTREADWRITESET</code>	(in module <i>hfc.protos.ledger.rwset.rwset_pb2</i>), 120
<code>__LAYOUT_QUANTITIESBYGROUPENTRY</code>	(in module <i>hfc.protos.discovery.protocol_pb2</i>), 109	<code>__NSREADWRITESET</code>	(in module <i>hfc.protos.ledger.rwset.rwset_pb2</i>), 120
<code>__LEADERSHIPMESSAGE</code>	(in module <i>hfc.protos.gossip.message_pb2</i>), 112	<code>__NYMSIGNATURE</code>	(in module <i>hfc.protos.idemix.idemix_pb2</i>), 117
		<code>__ORDERERADDRESSES</code>	(in module <i>hfc.protos.common.configuration_pb2</i>), 106
		<code>__ORGANIZATIONUNIT</code>	(in module <i>hfc.protos.msp.msp_principal_pb2</i>), 122
		<code>__PAYLOAD</code>	(in module <i>hfc.protos.msp.msp_principal_pb2</i>), 122

hfc.protos.common.common_pb2), 103
 _PAYLOAD (in module *hfc.protos.gossip.message_pb2*), 112
 _PEER (in module *hfc.protos.discovery.protocol_pb2*), 109
 _PEERENDPOINT (in module *hfc.protos.peer.peer_pb2*), 136
 _PEERID (in module *hfc.protos.peer.peer_pb2*), 136
 _PEERIDENTITY (in module *hfc.protos.gossip.message_pb2*), 112
 _PEERMEMBERSHIPQUERY (in module *hfc.protos.discovery.protocol_pb2*), 108
 _PEERMEMBERSHIPRESULT (in module *hfc.protos.discovery.protocol_pb2*), 108
 _PEERMEMBERSHIPRESULT_PEERSBYORENTRY (in module *hfc.protos.discovery.protocol_pb2*), 108
 _PEERS (in module *hfc.protos.discovery.protocol_pb2*), 109
 _PEERTIME (in module *hfc.protos.gossip.message_pb2*), 112
 _PLAINAPPROVE (in module *hfc.protos.token.transaction_pb2*), 145
 _PLAINDELEGATEDOUTPUT (in module *hfc.protos.token.transaction_pb2*), 145
 _PLAINEXPECTATION (in module *hfc.protos.token.expectations_pb2*), 142
 _PLAINIMPORT (in module *hfc.protos.token.transaction_pb2*), 145
 _PLAINOUTPUT (in module *hfc.protos.token.transaction_pb2*), 145
 _PLAINTOKENACTION (in module *hfc.protos.token.transaction_pb2*), 145
 _PLAINTOKENEXPECTATION (in module *hfc.protos.token.expectations_pb2*), 142
 _PLAINTRANSFER (in module *hfc.protos.token.transaction_pb2*), 145
 _PLAINTRANSFERFROM (in module *hfc.protos.token.transaction_pb2*), 145
 _POLICY (in module *hfc.protos.common.policies_pb2*), 107
 _POLICY_POLICYTYPE (in module *hfc.protos.common.policies_pb2*), 107
 _PRIVATE DATAMESSAGE (in module *hfc.protos.gossip.message_pb2*), 112
 _PRIVATEPAYLOAD (in module *hfc.protos.gossip.message_pb2*), 112
 _PROCESSEDTRANSACTION (in module *hfc.protos.peer.transaction_pb2*), 141
 _PROPERTIES (in module *hfc.protos.gossip.message_pb2*), 112
 _PROPOSAL (in module *hfc.protos.peer.proposal_pb2*), 137
 _PROPOSALRESPONSE (in module *hfc.protos.peer.proposal_response_pb2*), 138
 _PROPOSALRESPONSEPAYLOAD (in module *hfc.protos.peer.proposal_response_pb2*), 138
 _PROVER (in module *hfc.protos.token.prover_pb2*), 144
 _PULLMSGTYPE (in module *hfc.protos.gossip.message_pb2*), 112
 _PUTSTATE (in module *hfc.protos.peer.chaincode_shim_pb2*), 132
 _PUTSTATEMETADATA (in module *hfc.protos.peer.chaincode_shim_pb2*), 132
 _PVTDATADIGEST (in module *hfc.protos.gossip.message_pb2*), 113
 _PVTDATAELEMENT (in module *hfc.protos.gossip.message_pb2*), 113
 _PVTDATAPAYLOAD (in module *hfc.protos.gossip.message_pb2*), 113
 _QUERY (in module *hfc.protos.discovery.protocol_pb2*), 108
 _QUERYINSTALLEDCHAINCODEARGS (in module *hfc.protos.peer.lifecycle.lifecycle_pb2*), 128
 _QUERYINSTALLEDCHAINCODERESULT (in module *hfc.protos.peer.lifecycle.lifecycle_pb2*), 128
 _QUERYMETADATA (in module *hfc.protos.peer.chaincode_shim_pb2*), 132
 _QUERYREADS (in module *hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2*), 119
 _QUERYREADSMERKLESUMMARY (in module *hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2*), 119
 _QUERYRESPONSE (in module *hfc.protos.peer.chaincode_shim_pb2*), 132
 _QUERYRESPONSEMETADATA (in module *hfc.protos.peer.chaincode_shim_pb2*), 132
 _QUERYRESULT (in module *hfc.protos.discovery.protocol_pb2*), 108
 _QUERYRESULTBYTES (in module *hfc.protos.peer.chaincode_shim_pb2*), 132
 _QUERYSTATECLOSE (in module *hfc.protos.peer.chaincode_shim_pb2*), 132
 _QUERYSTATENEXT (in module *hfc.protos.peer.chaincode_shim_pb2*), 132
 _RANGEQUERYINFO (in module *hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2*), 119
 _RECIPIENTTRANSFERSHARE (in module *hfc.protos.token.prover_pb2*), 142
 _REDEEMREQUEST (in module *hfc.protos.token.prover_pb2*), 142
 _REMOTEPVTDATAREQUEST (in module *hfc.protos.gossip.message_pb2*), 113
 _REMOTEPVTDATA RESPONSE (in module *hfc.protos.gossip.message_pb2*), 113

<i>hfc.protos.gossip.message_pb2</i>), 113		<i>hfc.protos.discovery.protocol_pb2</i>), 108	
<code>_REMOTESTATEREQUEST</code> (in module	<i>hfc.protos.gossip.message_pb2</i>), 113	<code>_SIGNEDTRANSACTION</code> (in module	<i>hfc.protos.peer.transaction_pb2</i>), 141
<code>_REMOTESTATERESPONSE</code> (in module	<i>hfc.protos.gossip.message_pb2</i>), 113	<code>_SIGNINGIDENTITYINFO</code> (in module	<i>hfc.protos.msp.msp_config_pb2</i>), 122
<code>_REQUEST</code> (in module	<i>hfc.protos.discovery.protocol_pb2</i>), 108	<code>_STATEINFO</code> (in module	<i>hfc.protos.gossip.message_pb2</i>), 112
<code>_RESPONSE</code> (in module	<i>hfc.protos.discovery.protocol_pb2</i>), 108	<code>_STATEINFOPULLREQUEST</code> (in module	<i>hfc.protos.gossip.message_pb2</i>), 112
<code>_RESPONSE</code> (in module	<i>hfc.protos.peer.proposal_response_pb2</i>), 138	<code>_STATEINFOSNAPSHOT</code> (in module	<i>hfc.protos.gossip.message_pb2</i>), 112
<code>_SECRET</code> (in module <i>hfc.protos.gossip.message_pb2</i>), 112		<code>_STATEMETADATA</code> (in module	<i>hfc.protos.peer.chaincode_shim_pb2</i>), 132
<code>_SECRETENVELOPE</code> (in module	<i>hfc.protos.gossip.message_pb2</i>), 112	<code>_STATEMETADATARESULT</code> (in module	<i>hfc.protos.peer.chaincode_shim_pb2</i>), 132
<code>_SEEKINFO</code> (in module <i>hfc.protos.orderer.ab_pb2</i>), 123		<code>_STATICCOLLECTIONCONFIG</code> (in module	<i>hfc.protos.common.collection_pb2</i>), 102
<code>_SEEKINFO_SEEKBEHAVIOR</code> (in module	<i>hfc.protos.orderer.ab_pb2</i>), 123	<code>_STATUS</code> (in module <i>hfc.protos.common.common_pb2</i>), 102	
<code>_SEEKNEWEST</code> (in module <i>hfc.protos.orderer.ab_pb2</i>), 123		<code>_STEPREQUEST</code> (in module	<i>hfc.protos.orderer.cluster_pb2</i>), 125
<code>_SEEKOLDEST</code> (in module <i>hfc.protos.orderer.ab_pb2</i>), 123		<code>_STEPRESPONSE</code> (in module	<i>hfc.protos.orderer.cluster_pb2</i>), 125
<code>_SEEKPOSITION</code> (in module	<i>hfc.protos.orderer.ab_pb2</i>), 123	<code>_SUBMITREQUEST</code> (in module	<i>hfc.protos.orderer.cluster_pb2</i>), 125
<code>_SEEKSPECIFIED</code> (in module	<i>hfc.protos.orderer.ab_pb2</i>), 123	<code>_SUBMITRESPONSE</code> (in module	<i>hfc.protos.orderer.cluster_pb2</i>), 125
<code>_SERIALIZEDIDEMIXIDENTITY</code> (in module	<i>hfc.protos.msp.identities_pb2</i>), 121	<code>_TOKENEXPECTATION</code> (in module	<i>hfc.protos.token.expectations_pb2</i>), 142
<code>_SERIALIZEDIDENTITY</code> (in module	<i>hfc.protos.msp.identities_pb2</i>), 121	<code>_TOKENOUTPUT</code> (in module	<i>hfc.protos.token.prover_pb2</i>), 142
<code>_SERVERSTATUS</code> (in module	<i>hfc.protos.peer.admin_pb2</i>), 128	<code>_TOKENTOISSUE</code> (in module	<i>hfc.protos.token.prover_pb2</i>), 142
<code>_SERVERSTATUS_STATUSCODE</code> (in module	<i>hfc.protos.peer.admin_pb2</i>), 128	<code>_TOKENTRANSACTION</code> (in module	<i>hfc.protos.token.transaction_pb2</i>), 145
<code>_SIGNATURE</code> (in module	<i>hfc.protos.idemix.idemix_pb2</i>), 117	<code>_TRANSACTION</code> (in module	<i>hfc.protos.peer.transaction_pb2</i>), 141
<code>_SIGNATUREHEADER</code> (in module	<i>hfc.protos.common.common_pb2</i>), 103	<code>_TRANSACTIONACTION</code> (in module	<i>hfc.protos.peer.transaction_pb2</i>), 141
<code>_SIGNATUREPOLICY</code> (in module	<i>hfc.protos.common.policies_pb2</i>), 107	<code>_TRANSFERREQUEST</code> (in module	<i>hfc.protos.token.prover_pb2</i>), 142
<code>_SIGNATUREPOLICYENVELOPE</code> (in module	<i>hfc.protos.common.policies_pb2</i>), 107	<code>_TXPVTREADWRITESET</code> (in module	<i>hfc.protos.ledger.rwset.rwset_pb2</i>), 120
<code>_SIGNATUREPOLICY_NOUTOF</code> (in module	<i>hfc.protos.common.policies_pb2</i>), 107	<code>_TXPVTREADWRITESETWITHCONFIGINFO</code> (in module <i>hfc.protos.transientstore.transientstore_pb2</i>), 147	
<code>_SIGNEDCHAINCODEDEPLOYMENTSPEC</code> (in module	<i>hfc.protos.peer.signed_cc_dep_spec_pb2</i>), 140	<code>_TXPVTREADWRITESETWITHCONFIGINFO_COLLECTIONCONFIGS</code> (in module <i>hfc.protos.transientstore.transientstore_pb2</i>), 146	
<code>_SIGNEDCOMMAND</code> (in module	<i>hfc.protos.token.prover_pb2</i>), 142	<code>_TXREADWRITESET</code> (in module	<i>hfc.protos.ledger.rwset.rwset_pb2</i>), 120
<code>_SIGNEDCOMMANDRESPONSE</code> (in module	<i>hfc.protos.token.prover_pb2</i>), 143	<code>_TXREADWRITESET_DATAMODEL</code> (in module	<i>hfc.protos.ledger.rwset.rwset_pb2</i>), 120
<code>_SIGNEDPROPOSAL</code> (in module	<i>hfc.protos.peer.proposal_pb2</i>), 137	<code>_TXVALIDATIONCODE</code> (in module	
<code>_SIGNEDREQUEST</code> (in module			

hfc.protos.peer.transaction_pb2), 140
 __UNSPENTTOKENS (in module *hfc.protos.token.prover_pb2*), 142
 __VERSION (in module *hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2*), 119
 __VSCCARGS (in module *hfc.protos.peer.resources_pb2*), 139
 __enter__() (*hfc.util.utils.zeroTimeContextManager* method), 159
 __exit__() (*hfc.util.utils.zeroTimeContextManager* method), 159
 __init_internal_channel() (*hfc.fabric_network.network.Network* method), 100
 __str__() (*hfc.fabric.orderer.Orderer* method), 71
 __str__() (*hfc.fabric.peer.Peer* method), 74
 __str__() (*hfc.fabric.transaction.tx_context.TXContext* method), 48
 __str__() (*hfc.fabric.user.User* method), 76
 __str__() (*hfc.fabric_ca.caservice.Enrollment* method), 89
 __str__() (*hfc.util.keyvaluestore.FileKeyValueStore* method), 154
 _b (in module *hfc.protos.common.collection_pb2*), 101
 _b (in module *hfc.protos.common.common_pb2*), 102
 _b (in module *hfc.protos.common.configtx_pb2*), 104
 _b (in module *hfc.protos.common.configuration_pb2*), 106
 _b (in module *hfc.protos.common.ledger_pb2*), 107
 _b (in module *hfc.protos.common.policies_pb2*), 107
 _b (in module *hfc.protos.discovery.protocol_pb2*), 108
 _b (in module *hfc.protos.gossip.message_pb2*), 112
 _b (in module *hfc.protos.idemix.idemix_pb2*), 117
 _b (in module *hfc.protos.ledger.queryresult.kv_query_result_pb2*), 118
 _b (in module *hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2*), 119
 _b (in module *hfc.protos.ledger.rwset.rwset_pb2*), 120
 _b (in module *hfc.protos.msp.identities_pb2*), 121
 _b (in module *hfc.protos.msp.msp_config_pb2*), 121
 _b (in module *hfc.protos.msp.msp_principal_pb2*), 122
 _b (in module *hfc.protos.orderer.ab_pb2*), 123
 _b (in module *hfc.protos.orderer.cluster_pb2*), 125
 _b (in module *hfc.protos.orderer.configuration_pb2*), 126
 _b (in module *hfc.protos.orderer.kafka_pb2*), 127
 _b (in module *hfc.protos.peer.admin_pb2*), 128
 _b (in module *hfc.protos.peer.chaincode_event_pb2*), 130
 _b (in module *hfc.protos.peer.chaincode_pb2*), 130
 _b (in module *hfc.protos.peer.chaincode_shim_pb2*), 131
 _b (in module *hfc.protos.peer.configuration_pb2*), 134
 _b (in module *hfc.protos.peer.events_pb2*), 134
 _b (in module *hfc.protos.peer.lifecycle.lifecycle_pb2*), 128
 _b (in module *hfc.protos.peer.peer_pb2*), 136
 _b (in module *hfc.protos.peer.policy_pb2*), 137
 _b (in module *hfc.protos.peer.proposal_pb2*), 137
 _b (in module *hfc.protos.peer.proposal_response_pb2*), 138
 _b (in module *hfc.protos.peer.query_pb2*), 138
 _b (in module *hfc.protos.peer.resources_pb2*), 139
 _b (in module *hfc.protos.peer.signed_cc_dep_spec_pb2*), 139
 _b (in module *hfc.protos.peer.transaction_pb2*), 140
 _b (in module *hfc.protos.token.expectations_pb2*), 141
 _b (in module *hfc.protos.token.prover_pb2*), 142
 _b (in module *hfc.protos.token.transaction_pb2*), 145
 _b (in module *hfc.protos.transientstore.transientstore_pb2*), 146
 _build_channel_header() (*hfc.fabric.channel.channel.Channel* method), 38
 _build_policy() (*hfc.fabric.channel.channel.Channel* method), 39
 _build_principal() (*hfc.fabric.channel.channel.Channel* method), 39
 _build_proto_cc_interest() (*hfc.fabric.channel.channel.Channel* method), 41
 _check_malleability() (*hfc.util.crypto.crypto.Ecies* method), 150
 _check_policy() (*hfc.fabric.channel.channel.Channel* method), 39
 _create_instantiation_proposal() (in module *hfc.fabric.channel.instantiation*), 44
 _create_invocation_proposal() (in module *hfc.fabric.channel.invocation*), 46
 _create_or_update_channel() (*hfc.fabric.Client* method), 79
 _create_or_update_channel() (*hfc.fabric.client.Client* method), 64
 _create_or_update_channel_request() (*hfc.fabric.Client* method), 80
 _create_or_update_channel_request() (*hfc.fabric.client.Client* method), 64
 _create_seek_info() (*hfc.fabric.channel.channel_eventhub.ChannelEventHub* method), 43
 _discovery() (*hfc.fabric.channel.channel.Channel* method), 41
 _get_latest_block() (*hfc.fabric.channel.channel.Channel* method), 37
 _get_policy() (*hfc.fabric.channel.channel.Channel* method), 39
 _get_random_orderer() (*hfc.fabric.channel.channel.Channel* method),

hfc.protos.token.expectations_pb2), 142
 _options (in module *hfc.protos.token.prover_pb2*), 144
 _options (in module *hfc.protos.token.transaction_pb2*), 146
 _options (in module *hfc.protos.transientstore.transientstore_pb2*), 147
 _prevent_malleability() (*hfc.util.crypto.crypto.Ecies* method), 150
 _processBlockEvents() (*hfc.fabric.channel.channel_eventhub.ChannelEventHub* method), 43
 _processChaincodeEvents() (*hfc.fabric.channel.channel_eventhub.ChannelEventHub* method), 43
 _processTxEvents() (*hfc.fabric.channel.channel_eventhub.ChannelEventHub* method), 43
 _process_discovery_membership_result() (*hfc.fabric.Client* method), 86
 _process_discovery_membership_result() (*hfc.fabric.client.Client* method), 70
 _queue_chaincode_event() (*hfc.fabric.channel.channel_eventhub.ChannelEventHub* method), 43
 _restore_state() (*hfc.fabric.user.User* method), 76
 _save_state() (*hfc.fabric.user.User* method), 76
 _send_ca_delete() (*hfc.fabric_ca.CAClient* method), 95
 _send_ca_delete() (*hfc.fabric_ca.caservice.CAClient* method), 90
 _send_ca_get() (*hfc.fabric_ca.CAClient* method), 95
 _send_ca_get() (*hfc.fabric_ca.caservice.CAClient* method), 89
 _send_ca_post() (*hfc.fabric_ca.CAClient* method), 95
 _send_ca_post() (*hfc.fabric_ca.caservice.CAClient* method), 89
 _send_ca_update() (*hfc.fabric_ca.CAClient* method), 96
 _send_ca_update() (*hfc.fabric_ca.caservice.CAClient* method), 90
 _send_cc_proposal() (*hfc.fabric.channel.channel.Channel* method), 39
 _send_tx_proposal() (*hfc.fabric.channel.channel.Channel* static method), 39
 _sym_db (in module *hfc.protos.common.collection_pb2*), 101
 _sym_db (in module *hfc.protos.common.common_pb2*), 102
 _sym_db (in module *hfc.protos.common.configtx_pb2*), 104
 _sym_db (in module *hfc.protos.common.configuration_pb2*), 106
 _sym_db (in module *hfc.protos.common.ledger_pb2*), 107
 _sym_db (in module *hfc.protos.common.policies_pb2*), 107
 _sym_db (in module *hfc.protos.discovery.protocol_pb2*), 108
 _sym_db (in module *hfc.protos.gossip.message_pb2*), 112
 _sym_db (in module *hfc.protos.idemix.idemix_pb2*), 117
 _sym_db (in module *hfc.protos.ledger.queryresult.kv_query_result_pb2*), 118
 _sym_db (in module *hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2*), 119
 _sym_db (in module *hfc.protos.ledger.rwset.rwset_pb2*), 120
 _sym_db (in module *hfc.protos.msp.identities_pb2*), 121
 _sym_db (in module *hfc.protos.msp.msp_config_pb2*), 121
 _sym_db (in module *hfc.protos.msp.msp_principal_pb2*), 122
 _sym_db (in module *hfc.protos.orderer.ab_pb2*), 123
 _sym_db (in module *hfc.protos.orderer.cluster_pb2*), 125
 _sym_db (in module *hfc.protos.orderer.configuration_pb2*), 126
 _sym_db (in module *hfc.protos.orderer.kafka_pb2*), 127
 _sym_db (in module *hfc.protos.peer.admin_pb2*), 128
 _sym_db (in module *hfc.protos.peer.chaincode_event_pb2*), 130
 _sym_db (in module *hfc.protos.peer.chaincode_pb2*), 130
 _sym_db (in module *hfc.protos.peer.chaincode_shim_pb2*), 131
 _sym_db (in module *hfc.protos.peer.configuration_pb2*), 134
 _sym_db (in module *hfc.protos.peer.events_pb2*), 134
 _sym_db (in module *hfc.protos.peer.lifecycle.lifecycle_pb2*), 128
 _sym_db (in module *hfc.protos.peer.peer_pb2*), 136
 _sym_db (in module *hfc.protos.peer.policy_pb2*), 137
 _sym_db (in module *hfc.protos.peer.proposal_pb2*), 137
 _sym_db (in module *hfc.protos.peer.proposal_response_pb2*), 138
 _sym_db (in module *hfc.protos.peer.query_pb2*), 138
 _sym_db (in module *hfc.protos.peer.resources_pb2*), 139

- `_sym_db` (in module `hfc.protos.peer.signed_cc_dep_spec_pb2`), 139
- `_sym_db` (in module `hfc.protos.peer.transaction_pb2`), 140
- `_sym_db` (in module `hfc.protos.token.expectations_pb2`), 141
- `_sym_db` (in module `hfc.protos.token.prover_pb2`), 142
- `_sym_db` (in module `hfc.protos.token.transaction_pb2`), 145
- `_sym_db` (in module `hfc.protos.transientstore.transientstore_pb2`), 146
- `_validate_peer()` (`hfc.fabric.channel.channel.Channel` method), 37
- `_validate_peers()` (`hfc.fabric.channel.channel.Channel` method), 38
- `_validate_request()` (`hfc.fabric.Client` method), 80
- `_validate_request()` (`hfc.fabric.client.Client` method), 64
- `_validate_state()` (`hfc.fabric.channel.channel.Channel` method), 37
- ## A
- `account()` (`hfc.fabric.user.User` property), 75
- `Acknowledgement` (in module `hfc.protos.gossip.message_pb2`), 116
- ACLs (in module `hfc.protos.peer.configuration_pb2`), 134
- `add_AdminServicer_to_server()` (in module `hfc.protos.peer.admin_pb2_grpc`), 130
- `add_AtomicBroadcastServicer_to_server()` (in module `hfc.protos.orderer.ab_pb2_grpc`), 125
- `add_ChaincodeSupportServicer_to_server()` (in module `hfc.protos.peer.chaincode_shim_pb2_grpc`), 133
- `add_ClusterServicer_to_server()` (in module `hfc.protos.orderer.cluster_pb2_grpc`), 126
- `add_DeliverServicer_to_server()` (in module `hfc.protos.peer.events_pb2_grpc`), 136
- `add_DiscoveryServicer_to_server()` (in module `hfc.protos.discovery.protocol_pb2_grpc`), 111
- `add_EndorserServicer_to_server()` (in module `hfc.protos.peer.peer_pb2_grpc`), 136
- `add_GossipServicer_to_server()` (in module `hfc.protos.gossip.message_pb2_grpc`), 116
- `add_orderer()` (`hfc.fabric.channel.channel.Channel` method), 36
- `add_peer()` (`hfc.fabric.channel.channel.Channel` method), 37
- `add_ProverServicer_to_server()` (in module `hfc.protos.token.prover_pb2_grpc`), 145
- `AdminOperation` (in module `hfc.protos.peer.admin_pb2`), 129
- `AdminServicer` (class in `hfc.protos.peer.admin_pb2_grpc`), 129
- `AdminStub` (class in `hfc.protos.peer.admin_pb2_grpc`), 129
- `AES_KEY_LENGTH` (in module `hfc.util.crypto.crypto`), 148
- `affiliation()` (`hfc.fabric.user.User` property), 75
- `AffiliationService` (class in `hfc.fabric_ca.affiliationService`), 86
- `AliveMessage` (in module `hfc.protos.gossip.message_pb2`), 115
- `AllowanceRecipientShare` (in module `hfc.protos.token.prover_pb2`), 144
- `AnchorPeer` (in module `hfc.protos.peer.configuration_pb2`), 134
- `AnchorPeers` (in module `hfc.protos.peer.configuration_pb2`), 134
- `APIResource` (in module `hfc.protos.peer.configuration_pb2`), 134
- `ApplicationPolicy` (in module `hfc.protos.peer.policy_pb2`), 137
- `ApproveRequest` (in module `hfc.protos.token.prover_pb2`), 144
- `args()` (`hfc.fabric.transaction.tx_proposal_request.TXProposalRequest` property), 51
- `AsymmetricKey` (class in `hfc.util.crypto.crypto`), 149
- `async_get_value()` (`hfc.util.keyvaluestore.FileKeyValueStore` method), 154
- `async_get_value()` (`hfc.util.keyvaluestore.KeyValueStore` method), 153
- `async_set_value()` (`hfc.util.keyvaluestore.FileKeyValueStore` method), 154
- `async_set_value()` (`hfc.util.keyvaluestore.KeyValueStore` method), 153
- `AtomicBroadcastServicer` (class in `hfc.protos.orderer.ab_pb2_grpc`), 125
- `AtomicBroadcastStub` (class in `hfc.protos.orderer.ab_pb2_grpc`), 125
- `AuthInfo` (in module `hfc.protos.discovery.protocol_pb2`), 110
- ## B
- `BAD_CHANNEL_HEADER` (in module `hfc.protos.peer.transaction_pb2`), 141
- `BAD_COMMON_HEADER` (in module `hfc.protos.peer.transaction_pb2`), 140

BAD_CREATOR_SIGNATURE (in module *hfc.protos.peer.transaction_pb2*), 140
 BAD_HEADER_EXTENSION (in module *hfc.protos.peer.transaction_pb2*), 140
 BAD_PAYLOAD (in module *hfc.protos.peer.transaction_pb2*), 140
 BAD_PROPOSAL_TXID (in module *hfc.protos.peer.transaction_pb2*), 140
 BAD_REQUEST (in module *hfc.protos.common.common_pb2*), 102
 BAD_RESPONSE_PAYLOAD (in module *hfc.protos.peer.transaction_pb2*), 141
 BAD_RWSET (in module *hfc.protos.peer.transaction_pb2*), 141
 BatchSize (in module *hfc.protos.orderer.configuration_pb2*), 126
 BatchTimeout (in module *hfc.protos.orderer.configuration_pb2*), 126
 Block (in module *hfc.protos.common.common_pb2*), 104
 BLOCK_MSG (in module *hfc.protos.gossip.message_pb2*), 112
 BlockchainInfo (in module *hfc.protos.common.ledger_pb2*), 107
 BlockData (in module *hfc.protos.common.common_pb2*), 104
 BlockDataHashingStructure (in module *hfc.protos.common.configuration_pb2*), 106
 BlockDecoder (class in *hfc.fabric.block_decoder*), 54
 BlockHeader (in module *hfc.protos.common.common_pb2*), 104
 BlockMetadata (in module *hfc.protos.common.common_pb2*), 104
 BlockMetadataIndex (in module *hfc.protos.common.common_pb2*), 102
 broadcast () (*hfc.fabric.orderer.Orderer* method), 71
 Broadcast () (*hfc.protos.orderer.ab_pb2_grpc.AtomicBroadcastService* method), 125
 BroadcastResponse (in module *hfc.protos.orderer.ab_pb2*), 124
 build_cc_proposal () (in module *hfc.util.utils*), 157
 build_channel_header () (in module *hfc.util.utils*), 157
 build_header () (in module *hfc.util.utils*), 157
 build_tx_req () (in module *hfc.util.utils*), 159
C
 ca_service () (in module *hfc.fabric_ca.caservice*), 92
 caCert () (*hfc.fabric_ca.caservice.Enrollment* property), 89
 CAClient (class in *hfc.fabric_ca*), 95
 CAClient (class in *hfc.fabric_ca.caservice*), 89
 Capabilities (in module *hfc.protos.common.configuration_pb2*), 106
 Capability (in module *hfc.protos.common.configuration_pb2*), 106
 CAs () (*hfc.fabric.Client* property), 78
 CAs () (*hfc.fabric.client.Client* property), 62
 CAService (class in *hfc.fabric_ca.caservice*), 90
 cc_endorsement_policy () (*hfc.fabric.transaction.tx_proposal_request.TXProposalRequest* property), 51
 CC_INSTALL (in module *hfc.fabric.transaction.tx_proposal_request*), 50
 CC_INSTALL (in module *hfc.util.utils*), 156
 CC_INSTANTIATE (in module *hfc.fabric.transaction.tx_proposal_request*), 50
 CC_INVOKE (in module *hfc.fabric.transaction.tx_proposal_request*), 50
 cc_name () (*hfc.fabric.transaction.tx_proposal_request.TXProposalRequest* property), 50
 cc_path () (*hfc.fabric.transaction.tx_proposal_request.TXProposalRequest* property), 50
 CC_QUERY (in module *hfc.fabric.transaction.tx_proposal_request*), 50
 cc_type () (*hfc.fabric.transaction.tx_proposal_request.TXProposalRequest* property), 50
 CC_TYPE_CAR (in module *hfc.fabric.transaction.tx_proposal_request*), 50
 CC_TYPE_GOLANG (in module *hfc.fabric.transaction.tx_proposal_request*), 50
 CC_TYPE_GOLANG (in module *hfc.util.utils*), 156
 CC_TYPE_JAVA (in module *hfc.fabric.transaction.tx_proposal_request*), 50
 CC_TYPE_NODE (in module *hfc.fabric.transaction.tx_proposal_request*), 50
 CC_UPGRADE (in module *hfc.fabric.transaction.tx_proposal_request*), 50
 CDSData (in module *hfc.protos.peer.chaincode_pb2*), 131
 cert () (*hfc.fabric_ca.caservice.Enrollment* property), 89
 certificateAuthority (class in *hfc.fabric.certificateAuthority*), 61
 CertificateService (class in *hfc.fabric_ca.certificateService*), 93
 Chaincode (in module *hfc.protos.gossip.message_pb2*), 116
 chaincode_install () (*hfc.fabric.Client* method), 81
 chaincode_install () (*hfc.fabric.client.Client* method), 66
 chaincode_instantiate () (*hfc.fabric.Client* method), 81

`chaincode_instantiate()` (*hfc.fabric.client.Client method*), 66
`chaincode_instantiation()` (*in module hfc.fabric.channel.instantiation*), 45
`chaincode_invocation()` (*in module hfc.fabric.channel.invocation*), 47
`chaincode_invoke()` (*hfc.fabric.Client method*), 83
`chaincode_invoke()` (*hfc.fabric.client.Client method*), 67
`CHAINCODE_PACKAGE` (*in module hfc.protos.common.common_pb2*), 103
`chaincode_query()` (*hfc.fabric.Client method*), 83
`chaincode_query()` (*hfc.fabric.client.Client method*), 68
`chaincode_upgrade()` (*hfc.fabric.Client method*), 82
`chaincode_upgrade()` (*hfc.fabric.client.Client method*), 67
`CHAINCODE_VERSION_CONFLICT` (*in module hfc.protos.peer.transaction_pb2*), 140
`ChaincodeAction` (*in module hfc.protos.peer.proposal_pb2*), 138
`ChaincodeActionPayload` (*in module hfc.protos.peer.transaction_pb2*), 141
`ChaincodeCall` (*in module hfc.protos.discovery.protocol_pb2*), 110
`ChaincodeData` (*in module hfc.protos.peer.chaincode_pb2*), 131
`ChaincodeDeploymentSpec` (*in module hfc.protos.peer.chaincode_pb2*), 131
`ChaincodeEndorsedAction` (*in module hfc.protos.peer.transaction_pb2*), 141
`ChaincodeEndorsement` (*in module hfc.protos.peer.resources_pb2*), 139
`ChaincodeEvent` (*in module hfc.protos.peer.chaincode_event_pb2*), 130
`ChaincodeHeaderExtension` (*in module hfc.protos.peer.proposal_pb2*), 137
`ChaincodeID` (*in module hfc.protos.peer.chaincode_pb2*), 131
`ChaincodeIdentifier` (*in module hfc.protos.peer.resources_pb2*), 139
`ChaincodeInfo` (*in module hfc.protos.peer.query_pb2*), 139
`ChaincodeInput` (*in module hfc.protos.peer.chaincode_pb2*), 131
`ChaincodeInterest` (*in module hfc.protos.discovery.protocol_pb2*), 110
`ChaincodeInvocationSpec` (*in module hfc.protos.peer.chaincode_pb2*), 131
`ChaincodeMessage` (*in module hfc.protos.peer.chaincode_shim_pb2*), 132
`ChaincodeProposalPayload` (*in module hfc.protos.peer.proposal_pb2*), 137
`ChaincodeQuery` (*in module hfc.protos.discovery.protocol_pb2*), 110
`ChaincodeQueryResponse` (*in module hfc.protos.peer.query_pb2*), 139
`ChaincodeQueryResult` (*in module hfc.protos.discovery.protocol_pb2*), 110
`ChaincodeRegistration` (*class in hfc.fabric.channel.channel_eventhub*), 42
`ChaincodeSpec` (*in module hfc.protos.peer.chaincode_pb2*), 131
`ChaincodeSupportServicer` (*class in hfc.protos.peer.chaincode_shim_pb2_grpc*), 133
`ChaincodeSupportStub` (*class in hfc.protos.peer.chaincode_shim_pb2_grpc*), 133
`ChaincodeValidation` (*in module hfc.protos.peer.resources_pb2*), 139
`Channel` (*class in hfc.fabric.channel.channel*), 36
`channel_create()` (*hfc.fabric.Client method*), 78
`channel_create()` (*hfc.fabric.client.Client method*), 63
`channel_join()` (*hfc.fabric.Client method*), 78
`channel_join()` (*hfc.fabric.client.Client method*), 63
`channel_signconfigtx()` (*hfc.fabric.Client method*), 79
`channel_signconfigtx()` (*hfc.fabric.client.Client method*), 64
`channel_update()` (*hfc.fabric.Client method*), 78
`channel_update()` (*hfc.fabric.client.Client method*), 63
`ChannelConfiguration` (*class in hfc.fabric.channel.channel_configuration*), 42
`ChannelEventHub` (*class in hfc.fabric.channel.channel_eventhub*), 42
`ChannelHeader` (*in module hfc.protos.common.common_pb2*), 104
`ChannelInfo` (*in module hfc.protos.peer.query_pb2*), 139
`ChannelQueryResponse` (*in module hfc.protos.peer.query_pb2*), 139
`ChannelRestrictions` (*in module hfc.protos.orderer.configuration_pb2*), 127
`channels()` (*hfc.fabric.peer.Peer property*), 74
`check_replay_end()` (*hfc.fabric.channel.channel_eventhub.ChannelEventHub method*), 43
`check_start_stop_connect()` (*hfc.fabric.channel.channel_eventhub.ChannelEventHub method*), 43
`check_start_stop_listener()` (*hfc.fabric.channel.channel_eventhub.ChannelEventHub method*), 43

Client (class in *hfc.fabric*), 76
 Client (class in *hfc.fabric.client*), 61
 close_grpc_channels() (*hfc.fabric.Client* method), 77
 close_grpc_channels() (*hfc.fabric.client.Client* method), 61
 ClusterServicer (class in *hfc.protos.orderer.cluster_pb2_grpc*), 126
 ClusterStub (class in *hfc.protos.orderer.cluster_pb2_grpc*), 126
 CollectionConfig (in module *hfc.protos.common.collection_pb2*), 102
 CollectionConfigPackage (in module *hfc.protos.common.collection_pb2*), 102
 CollectionCriteria (in module *hfc.protos.common.collection_pb2*), 102
 CollectionHashedReadWriteSet (in module *hfc.protos.ledger.rwset.rwset_pb2*), 121
 CollectionPolicyConfig (in module *hfc.protos.common.collection_pb2*), 102
 CollectionPvtReadWriteSet (in module *hfc.protos.ledger.rwset.rwset_pb2*), 121
 collections_config() (*hfc.fabric.transaction.tx_proposal_request.TXProposalRequest* property), 51
 CombinedPrincipal (in module *hfc.protos.msp.msp_principal_pb2*), 123
 Command (in module *hfc.protos.token.prover_pb2*), 144
 CommandResponse (in module *hfc.protos.token.prover_pb2*), 144
 CommandResponseHeader (in module *hfc.protos.token.prover_pb2*), 144
 CONFIDENTIAL (in module *hfc.protos.peer.chaincode_pb2*), 130
 ConfidentialityLevel (in module *hfc.protos.peer.chaincode_pb2*), 130
 CONFIG (in module *hfc.protos.common.common_pb2*), 103
 Config (in module *hfc.protos.common.configtx_pb2*), 105
 config() (*hfc.fabric.channel.channel_configuration.ChannelConfiguration* property), 42
 CONFIG_UPDATE (in module *hfc.protos.common.common_pb2*), 103
 ConfigEnvelope (in module *hfc.protos.common.configtx_pb2*), 105
 ConfigGroup (in module *hfc.protos.common.configtx_pb2*), 105
 ConfigGroupSchema (in module *hfc.protos.common.configtx_pb2*), 105
 ConfigPolicy (in module *hfc.protos.common.configtx_pb2*), 106
 ConfigPolicySchema (in module *hfc.protos.common.configtx_pb2*), 105
 ConfigQuery (in module *hfc.protos.discovery.protocol_pb2*), 110
 ConfigResult (in module *hfc.protos.discovery.protocol_pb2*), 110
 ConfigSignature (in module *hfc.protos.common.configtx_pb2*), 106
 ConfigTree (in module *hfc.protos.peer.resources_pb2*), 139
 ConfigUpdate (in module *hfc.protos.common.configtx_pb2*), 105
 ConfigUpdateEnvelope (in module *hfc.protos.common.configtx_pb2*), 105
 ConfigValue (in module *hfc.protos.common.configtx_pb2*), 106
 ConfigValueSchema (in module *hfc.protos.common.configtx_pb2*), 105
 connect() (*hfc.fabric.channel.channel_eventhub.ChannelEventHub* method), 43
 connect() (*hfc.fabric_network.gateway.Gateway* method), 98
 connected() (*hfc.fabric.channel.channel_eventhub.ChannelEventHub* property), 43
 ConnEstablish (in module *hfc.protos.gossip.message_pb2*), 115
 ConsensusType (in module *hfc.protos.orderer.configuration_pb2*), 126
 consoleHandler (in module *hfc.fabric.client*), 61
 consoleHandler (in module *hfc.fabric_network.contract*), 97
 consoleHandler (in module *hfc.fabric_network.gateway*), 98
 consoleHandler (in module *hfc.fabric_network.network*), 100
 Consortium (in module *hfc.protos.common.configuration_pb2*), 106
 containing_oneof (in module *hfc.protos.common.collection_pb2*), 102
 containing_oneof (in module *hfc.protos.common.policies_pb2*), 107
 containing_oneof (in module *hfc.protos.discovery.protocol_pb2*), 109
 containing_oneof (in module *hfc.protos.gossip.message_pb2*), 113, 114
 containing_oneof (in module *hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2*), 119
 containing_oneof (in module *hfc.protos.orderer.ab_pb2*), 124
 containing_oneof (in module *hfc.protos.orderer.kafka_pb2*), 127
 containing_oneof (in module *hfc.protos.peer.admin_pb2*), 129
 containing_oneof (in module *hfc.protos.peer.events_pb2*), 135

containing_oneof (in module *hfc.protos.peer.policy_pb2*), 137
 containing_oneof (in module *hfc.protos.token.expectations_pb2*), 142
 containing_oneof (in module *hfc.protos.token.prover_pb2*), 143, 144
 containing_oneof (in module *hfc.protos.token.transaction_pb2*), 145, 146
 containing_type (in module *hfc.protos.common.configtx_pb2*), 105
 containing_type (in module *hfc.protos.common.configuration_pb2*), 106
 containing_type (in module *hfc.protos.common.policies_pb2*), 107, 108
 containing_type (in module *hfc.protos.discovery.protocol_pb2*), 109, 110
 containing_type (in module *hfc.protos.gossip.message_pb2*), 114
 containing_type (in module *hfc.protos.ledger.rwset.rwset_pb2*), 120
 containing_type (in module *hfc.protos.msp.msp_principal_pb2*), 123
 containing_type (in module *hfc.protos.orderer.ab_pb2*), 124
 containing_type (in module *hfc.protos.orderer.kafka_pb2*), 127
 containing_type (in module *hfc.protos.peer.admin_pb2*), 129
 containing_type (in module *hfc.protos.peer.chaincode_pb2*), 131
 containing_type (in module *hfc.protos.peer.chaincode_shim_pb2*), 132
 containing_type (in module *hfc.protos.peer.configuration_pb2*), 134
 containing_type (in module *hfc.protos.peer.proposal_pb2*), 137
 containing_type (in module *hfc.protos.transientstore.transientstore_pb2*), 147
 Contract (class in *hfc.fabric_network.contract*), 97
 convert_to_string() (*hfc.fabric.block_decoder.HeaderType* static method), 55
 CouchDBWalletStore (class in *hfc.fabric_network.couchdbwalletstore*), 97
 create() (*hfc.fabric_ca.affiliationService.AffiliationService* method), 86
 create() (*hfc.fabric_ca.identityService.IdentityService* method), 94
 create_app_channel() (in module *hfc.fabric.channel.channel*), 41
 create_ca() (in module *hfc.fabric.certificateAuthority*), 61
 create_cc_spec() (in module *hfc.protos.utils*), 147
 create_envelope() (in module *hfc.protos.utils*), 147
 create_grpc_channel() (in module *hfc.util.channel*), 152
 create_instantiation_proposal_req() (in module *hfc.fabric.channel.instantiation*), 45
 create_invocation_proposal_req() (in module *hfc.fabric.channel.invocation*), 46
 create_onCcEvent() (*hfc.fabric.Client* method), 81
 create_onCcEvent() (*hfc.fabric.client.Client* method), 66
 create_onCcEventArray() (*hfc.fabric.Client* method), 81
 create_onCcEventArray() (*hfc.fabric.client.Client* method), 66
 create_org() (in module *hfc.fabric.organization*), 73
 create_peer() (in module *hfc.fabric.peer*), 74
 create_seek_info() (in module *hfc.protos.utils*), 147
 create_seek_payload() (in module *hfc.protos.utils*), 147
 create_serialized_identity() (in module *hfc.util.utils*), 156
 create_system_channel() (in module *hfc.fabric.channel.channel*), 41
 create_tx_context() (in module *hfc.fabric.transaction.tx_context*), 49
 create_tx_payload() (in module *hfc.protos.utils*), 147
 create_tx_prop_req() (in module *hfc.fabric.transaction.tx_proposal_request*), 51
 create_user() (*hfc.fabric_network.couchdbwalletstore.CouchDBWalletStore* method), 98
 create_user() (*hfc.fabric_network.inmemorywalletstore.InMemoryWalletStore* method), 99
 create_user() (*hfc.fabric_network.wallet.FileSystemWallet* method), 101
 create_user() (in module *hfc.fabric.user*), 76
 CreateIdentity() (*hfc.fabric_network.wallet.Identity* method), 101
 Credential (in module *hfc.protos.idemix.idemix_pb2*), 118
 CredentialRevocationInformation (in module *hfc.protos.idemix.idemix_pb2*), 118
 CredRequest (in module *hfc.protos.idemix.idemix_pb2*), 118
 Crypto (class in *hfc.util.crypto.crypto*), 149
 crypto() (*hfc.fabric.transaction.tx_context.TXContext* property), 49
 crypto_suite() (*hfc.fabric.Client* property), 80
 crypto_suite() (*hfc.fabric.client.Client* property), 65

cryptoSuite() (*hfc.fabric.user.User* property), 75
 current_timestamp() (in module *hfc.util.utils*), 157
 CURVE_P_256_Size (in module *hfc.util.crypto.crypto*), 148
 CURVE_P_384_Size (in module *hfc.util.crypto.crypto*), 148

D

d2s (in module *hfc.util.policies*), 155
 dash() (*hfc.util.policies.String2Dict* method), 155
 DataDigest (in module *hfc.protos.gossip.message_pb2*), 115
 DataMessage (in module *hfc.protos.gossip.message_pb2*), 115
 DataRequest (in module *hfc.protos.gossip.message_pb2*), 115
 DataUpdate (in module *hfc.protos.gossip.message_pb2*), 115
 decode() (*hfc.fabric.block_decoder.BlockDecoder* static method), 54
 decode() (*hfc.fabric.block_decoder.FilteredBlockDecoder* static method), 54
 decode_block_data() (in module *hfc.fabric.block_decoder*), 55
 decode_block_data_envelope() (in module *hfc.fabric.block_decoder*), 55
 decode_block_header() (in module *hfc.fabric.block_decoder*), 55
 decode_block_metadata() (in module *hfc.fabric.block_decoder*), 55
 decode_chaincode_action() (in module *hfc.fabric.block_decoder*), 59
 decode_chaincode_action_payload() (in module *hfc.fabric.block_decoder*), 59
 decode_chaincode_endorsed_action() (in module *hfc.fabric.block_decoder*), 59
 decode_chaincode_events() (in module *hfc.fabric.block_decoder*), 59
 decode_chaincode_id() (in module *hfc.fabric.block_decoder*), 59
 decode_chaincode_proposal_payload() (in module *hfc.fabric.block_decoder*), 59
 decode_channel_header() (in module *hfc.fabric.block_decoder*), 55
 decode_config() (in module *hfc.fabric.block_decoder*), 56
 decode_config_envelope() (in module *hfc.fabric.block_decoder*), 56
 decode_config_group() (in module *hfc.fabric.block_decoder*), 57
 decode_config_groups() (in module *hfc.fabric.block_decoder*), 57
 decode_config_policies() (in module *hfc.fabric.block_decoder*), 57
 decode_config_policy() (in module *hfc.fabric.block_decoder*), 57
 decode_config_signature() (in module *hfc.fabric.block_decoder*), 58
 decode_config_update() (in module *hfc.fabric.block_decoder*), 57
 decode_config_update_envelope() (in module *hfc.fabric.block_decoder*), 56
 decode_config_value() (in module *hfc.fabric.block_decoder*), 57
 decode_config_values() (in module *hfc.fabric.block_decoder*), 57
 decode_crypto_config() (in module *hfc.fabric.block_decoder*), 58
 decode_endorsement() (in module *hfc.fabric.block_decoder*), 59
 decode_endorser_transaction() (in module *hfc.fabric.block_decoder*), 56
 decode_fabric_endpoints() (in module *hfc.fabric.block_decoder*), 60
 decode_fabric_MSP_config() (in module *hfc.fabric.block_decoder*), 58
 decode_fabric_Nodes_OUs() (in module *hfc.fabric.block_decoder*), 58
 decode_fabric_OU_identifier() (in module *hfc.fabric.block_decoder*), 58
 decode_fabric_peers_info() (in module *hfc.fabric.block_decoder*), 60
 decode_header() (in module *hfc.fabric.block_decoder*), 55
 decode_identity() (in module *hfc.fabric.block_decoder*), 56
 decode_implicit_meta_policy() (in module *hfc.fabric.block_decoder*), 57
 decode_key_info() (in module *hfc.fabric.block_decoder*), 58
 decode_kv_read() (in module *hfc.fabric.block_decoder*), 60
 decode_kv_rw_set() (in module *hfc.fabric.block_decoder*), 59
 decode_kv_write() (in module *hfc.fabric.block_decoder*), 60
 decode_last_config_sequence_number() (in module *hfc.fabric.block_decoder*), 56
 decode_metadata_signatures() (in module *hfc.fabric.block_decoder*), 56
 decode_metadata_value_signatures() (in module *hfc.fabric.block_decoder*), 56
 decode_MSP_principal() (in module *hfc.fabric.block_decoder*), 58
 decode_payload_based_on_type() (*hfc.fabric.block_decoder.HeaderType* static

method), 55
 decode_proposal_response_payload() (in module *hfc.fabric.block_decoder*), 59
 decode_range_query_info() (in module *hfc.fabric.block_decoder*), 60
 decode_readwrite_sets() (in module *hfc.fabric.block_decoder*), 59
 decode_response() (in module *hfc.fabric.block_decoder*), 60
 decode_signature_header() (in module *hfc.fabric.block_decoder*), 55
 decode_signature_policy() (in module *hfc.fabric.block_decoder*), 57
 decode_signature_policy_envelope() (in module *hfc.fabric.block_decoder*), 57
 decode_signing_identity_info() (in module *hfc.fabric.block_decoder*), 58
 decode_transaction() (*hfc.fabric.block_decoder.BlockDecoder* static method), 54
 decode_transaction_filter() (in module *hfc.fabric.block_decoder*), 56
 decode_version() (in module *hfc.fabric.block_decoder*), 55
 decrypt() (*hfc.util.crypto.crypto.Crypto* method), 149
 decrypt() (*hfc.util.crypto.crypto.Ecies* method), 151
 DEFAULT (in module *hfc.fabric.config.default*), 48
 DEFAULT_CA_BASE_URL (in module *hfc.fabric_ca.caservice*), 88
 DEFAULT_CA_ENDPOINT (in module *hfc.fabric_ca.caservice*), 88
 DEFAULT_NONCE_SIZE (in module *hfc.util.crypto.crypto*), 148
 DEFAULT_ORDERER_ENDPOINT (in module *hfc.fabric.orderer*), 71
 DEFAULT_PEER_ENDPOINT (in module *hfc.fabric.peer*), 73
 delete() (*hfc.fabric_ca.affiliationService.AffiliationService* method), 87
 delete() (*hfc.fabric_ca.identityService.IdentityService* method), 94
 Deliver() (*hfc.protos.orderer.ab_pb2_grpc.AtomicBroadcastService* method), 125
 Deliver() (*hfc.protos.peer.events_pb2_grpc.DeliverServicer* method), 135
 DELIVER_SEEK_INFO (in module *hfc.protos.common.common_pb2*), 103
 DeliverFiltered() (*hfc.protos.peer.events_pb2_grpc.DeliverServicer* method), 135
 DeliverResponse (in module *hfc.protos.orderer.ab_pb2*), 124
 DeliverResponse (in module *hfc.protos.peer.events_pb2*), 135
 DeliverServicer (class in *hfc.protos.peer.events_pb2_grpc*), 135
 DeliverStub (class in *hfc.protos.peer.events_pb2_grpc*), 135
 delivery() (*hfc.fabric.orderer.Orderer* method), 71
 delivery() (*hfc.fabric.peer.Peer* method), 74
 DelState (in module *hfc.protos.peer.chaincode_shim_pb2*), 132
 DESCRIPTOR (in module *hfc.protos.common.collection_pb2*), 101
 DESCRIPTOR (in module *hfc.protos.common.common_pb2*), 102
 DESCRIPTOR (in module *hfc.protos.common.configtx_pb2*), 104
 DESCRIPTOR (in module *hfc.protos.common.configuration_pb2*), 106
 DESCRIPTOR (in module *hfc.protos.common.ledger_pb2*), 107
 DESCRIPTOR (in module *hfc.protos.common.policies_pb2*), 107
 DESCRIPTOR (in module *hfc.protos.discovery.protocol_pb2*), 108
 DESCRIPTOR (in module *hfc.protos.gossip.message_pb2*), 112
 DESCRIPTOR (in module *hfc.protos.idemix.idemix_pb2*), 117
 DESCRIPTOR (in module *hfc.protos.ledger.queryresult.kv_query_result_pb2*), 118
 DESCRIPTOR (in module *hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2*), 119
 DESCRIPTOR (in module *hfc.protos.ledger.rwset.rwset_pb2*), 120
 DESCRIPTOR (in module *hfc.protos.msp.identities_pb2*), 121
 DESCRIPTOR (in module *hfc.protos.msp.msp_config_pb2*), 121
 DESCRIPTOR (in module *hfc.protos.msp.msp_principal_pb2*), 122
 DESCRIPTOR (in module *hfc.protos.orderer.ab_pb2*), 123
 DESCRIPTOR (in module *hfc.protos.orderer.cluster_pb2*), 125
 DESCRIPTOR (in module *hfc.protos.orderer.configuration_pb2*), 126
 DESCRIPTOR (in module *hfc.protos.orderer.kafka_pb2*), 127
 DESCRIPTOR (in module *hfc.protos.peer.admin_pb2*), 128
 DESCRIPTOR (in module *hfc.protos.peer.chaincode_event_pb2*), 130
 DESCRIPTOR (in module *hfc.protos.peer.chaincode_pb2*), 130

DESCRIPTOR (in module *hfc.protos.peer.chaincode_shim_pb2*), 131
 DESCRIPTOR (in module *hfc.protos.peer.configuration_pb2*), 134
 DESCRIPTOR (in module *hfc.protos.peer.events_pb2*), 134
 DESCRIPTOR (in module *hfc.protos.peer.lifecycle.lifecycle_pb2*), 128
 DESCRIPTOR (in module *hfc.protos.peer.peer_pb2*), 136
 DESCRIPTOR (in module *hfc.protos.peer.policy_pb2*), 137
 DESCRIPTOR (in module *hfc.protos.peer.proposal_pb2*), 137
 DESCRIPTOR (in module *hfc.protos.peer.proposal_response_pb2*), 138
 DESCRIPTOR (in module *hfc.protos.peer.query_pb2*), 138
 DESCRIPTOR (in module *hfc.protos.peer.resources_pb2*), 139
 DESCRIPTOR (in module *hfc.protos.peer.signed_cc_dep_spec_pb2*), 139
 DESCRIPTOR (in module *hfc.protos.peer.transaction_pb2*), 140
 DESCRIPTOR (in module *hfc.protos.token.expectations_pb2*), 142
 DESCRIPTOR (in module *hfc.protos.token.prover_pb2*), 142
 DESCRIPTOR (in module *hfc.protos.token.transaction_pb2*), 145
 DESCRIPTOR (in module *hfc.protos.transientstore.transientstore_pb2*), 146
 Dict2String (class in *hfc.util.policies*), 155
 disconnect () (*hfc.fabric.channel.channel_eventhub.ChannelEventHub* method), 43
 disconnect () (*hfc.fabric_network.gateway.Gateway* method), 99
 Discover () (*hfc.protos.discovery.protocol_pb2_grpc.DiscoveryService* method), 111
 DiscoveryServicer (class in *hfc.protos.discovery.protocol_pb2_grpc*), 111
 DiscoveryStub (class in *hfc.protos.discovery.protocol_pb2_grpc*), 111
 dot () (*hfc.util.policies.String2Dict* method), 155
 DUPLICATE_TXID (in module *hfc.protos.peer.transaction_pb2*), 140
E
 e () (*hfc.util.policies.String2Dict* method), 155
 Ecies (class in *hfc.util.crypto.crypto*), 150
 ecies () (in module *hfc.util.crypto.crypto*), 151
 ECP (in module *hfc.protos.idemix.idemix_pb2*), 117
 ECP2 (in module *hfc.protos.idemix.idemix_pb2*), 118
 emit () (*hfc.fabric.channel.NullHandler* method), 47
 emit () (*hfc.fabric.NullHandler* method), 86
 emit () (*hfc.fabric.transaction.NullHandler* method), 52
 emit () (*hfc.fabric_ca.NullHandler* method), 96
 emit () (*hfc.util.crypto.NullHandler* method), 152
 emit () (*hfc.util.NullHandler* method), 159
 Empty (in module *hfc.protos.gossip.message_pb2*), 115
 encrypt () (*hfc.util.crypto.crypto.Crypto* method), 149
 encrypt () (*hfc.util.crypto.crypto.Ecies* method), 151
 END_ONLY (in module *hfc.fabric.channel.channel_eventhub*), 42
 Endorsement (in module *hfc.protos.peer.proposal_response_pb2*), 138
 ENDORSEMENT_POLICY_FAILURE (in module *hfc.protos.peer.transaction_pb2*), 140
 EndorsementDescriptor (in module *hfc.protos.discovery.protocol_pb2*), 110
 ENDORSER_TRANSACTION (in module *hfc.protos.common.common_pb2*), 103
 EndorserServicer (class in *hfc.protos.peer.peer_pb2_grpc*), 136
 EndorserStub (class in *hfc.protos.peer.peer_pb2_grpc*), 136
 Endpoint (in module *hfc.protos.discovery.protocol_pb2*), 111
 endpoint () (*hfc.fabric.orderer.Orderer* property), 71
 endpoint () (*hfc.fabric.peer.Peer* property), 74
 Endpoints (in module *hfc.protos.discovery.protocol_pb2*), 110
 enroll () (*hfc.fabric_ca.CAClient* method), 96
 enroll () (*hfc.fabric_ca.caservice.CAClient* method), 90
 enroll () (*hfc.fabric_ca.caservice.CAService* method), 90
 EnrollmentService (class in *hfc.fabric_ca.caservice*), 88
 enrollment () (*hfc.fabric.user.User* property), 75
 enrollment_secret () (*hfc.fabric.user.User* property), 75
 enum_type (in module *hfc.protos.common.policies_pb2*), 108
 enum_type (in module *hfc.protos.gossip.message_pb2*), 113, 114
 enum_type (in module *hfc.protos.ledger.rwset.rwset_pb2*), 120
 enum_type (in module *hfc.protos.msp.msp_principal_pb2*), 123
 enum_type (in module *hfc.protos.orderer.ab_pb2*), 124
 enum_type (in module *hfc.protos.orderer.cluster_pb2*), 125

- enum_type (in module *hfc.protos.orderer.kafka_pb2*), 127
- enum_type (in module *hfc.protos.peer.admin_pb2*), 129
- enum_type (in module *hfc.protos.peer.chaincode_pb2*), 131
- enum_type (in module *hfc.protos.peer.chaincode_shim_pb2*), 132
- enum_type (in module *hfc.protos.peer.events_pb2*), 134, 135
- Envelope (in module *hfc.protos.common.common_pb2*), 104
- Envelope (in module *hfc.protos.gossip.message_pb2*), 115
- epoch() (*hfc.fabric.transaction.tx_context.TXContext* property), 48
- Error (in module *hfc.protos.discovery.protocol_pb2*), 110
- Error (in module *hfc.protos.token.prover_pb2*), 144
- evaluate_transaction() (*hfc.fabric_network.contract.Contract* method), 97
- EventRegistration (class in *hfc.fabric.channel.channel_eventhub*), 42
- exists() (*hfc.fabric_network.couchdbwalletstore.CouchDBWalletStore* method), 97
- exists() (*hfc.fabric_network.inmemorywalletstore.InMemoryWalletStore* method), 99
- exists() (*hfc.fabric_network.wallet.FileSystemWallet* method), 100
- ExpectationRequest (in module *hfc.protos.token.prover_pb2*), 144
- EXPIRED_CHAINCODE (in module *hfc.protos.peer.transaction_pb2*), 140
- export_net_profile() (*hfc.fabric.Client* method), 77
- export_net_profile() (*hfc.fabric.client.Client* method), 62
- extract_channel_config() (*hfc.fabric.Client* method), 79
- extract_channel_config() (*hfc.fabric.client.Client* method), 64
- extract_channel_config() (in module *hfc.util.utils*), 157
- ## F
- FabricCryptoConfig (in module *hfc.protos.msp.msp_config_pb2*), 122
- FabricMSPConfig (in module *hfc.protos.msp.msp_config_pb2*), 122
- FabricNodeOUs (in module *hfc.protos.msp.msp_config_pb2*), 122
- FabricOUIIdentifier (in module *hfc.protos.msp.msp_config_pb2*), 122
- fcn() (*hfc.fabric.transaction.tx_proposal_request.TXProposalRequest* property), 51
- file_key_value_store() (in module *hfc.util.keyvaluestore*), 154
- FileKeyValueStore (class in *hfc.util.keyvaluestore*), 153
- FileSystemWallet (class in *hfc.fabric_network.wallet*), 100
- FilteredBlock (in module *hfc.protos.peer.events_pb2*), 135
- FilteredBlockDecoder (class in *hfc.fabric.block_decoder*), 54
- FilteredChaincodeAction (in module *hfc.protos.peer.events_pb2*), 135
- FilteredTransaction (in module *hfc.protos.peer.events_pb2*), 135
- FilteredTransactionActions (in module *hfc.protos.peer.events_pb2*), 135
- FORBIDDEN (in module *hfc.protos.common.common_pb2*), 102
- ## G
- Gateway (class in *hfc.fabric_network.gateway*), 98
- generate_channel_tx() (*hfc.fabric.Client* method), 81
- generate_channel_tx() (*hfc.fabric.client.Client* method), 65
- generate_csr() (*hfc.util.crypto.crypto.Ecies* method), 151
- generate_nonce() (*hfc.util.crypto.crypto.Crypto* static method), 150
- generate_nonce() (in module *hfc.util.crypto.crypto*), 150
- generate_private_key() (*hfc.util.crypto.crypto.Crypto* method), 149
- generate_private_key() (*hfc.util.crypto.crypto.Ecies* method), 150
- generateAuthToken() (*hfc.fabric_ca.CAClient* method), 95
- generateAuthToken() (*hfc.fabric_ca.caservice.CAClient* method), 89
- generateCRL() (*hfc.fabric_ca.CAClient* method), 96
- generateCRL() (*hfc.fabric_ca.caservice.CAClient* method), 90
- generateCRL() (*hfc.fabric_ca.caservice.CAService* method), 92
- generateCRL() (*hfc.fabric_ca.caservice.Enrollment* method), 89
- get_attrs() (*hfc.fabric.orderer.Orderer* method), 71
- get_attrs() (*hfc.fabric.peer.Peer* method), 74
- get_attrs() (*hfc.fabric.transaction.tx_context.TXContext* method), 48
- get_attrs() (*hfc.fabric.user.User* method), 76

get_attrs () (*hfc.fabric_ca.caservice.Enrollment method*), 89
 get_attrs () (*hfc.util.keyvaluestore.FileKeyValueStore method*), 154
 get_block_between () (*hfc.fabric.channel.channel.Channel method*), 39
 get_cainfo () (*hfc.fabric_ca.CAClient method*), 96
 get_cainfo () (*hfc.fabric_ca.caservice.CAClient method*), 90
 get_cc_name () (*hfc.fabric_network.contract.Contract method*), 97
 get_channel () (*hfc.fabric.Client method*), 78
 get_channel () (*hfc.fabric.client.Client method*), 63
 get_channel_config () (*hfc.fabric.channel.channel.Channel method*), 40
 get_channel_config () (*hfc.fabric.Client method*), 79
 get_channel_config () (*hfc.fabric.client.Client method*), 63
 get_channel_config_with_orderer () (*hfc.fabric.channel.channel.Channel method*), 41
 get_channel_config_with_orderer () (*hfc.fabric.Client method*), 79
 get_channel_config_with_orderer () (*hfc.fabric.client.Client method*), 63
 get_client () (*hfc.fabric_network.gateway.Gateway method*), 99
 get_contract () (*hfc.fabric_network.network.Network method*), 100
 get_current_identity () (*hfc.fabric_network.gateway.Gateway method*), 98
 get_genesis_block () (*hfc.fabric.orderer.Orderer method*), 71
 get_logic () (*hfc.util.policies.String2Dict method*), 155
 get_net_info () (*hfc.fabric.Client method*), 77
 get_net_info () (*hfc.fabric.client.Client method*), 62
 get_network () (*hfc.fabric_network.contract.Contract method*), 97
 get_network () (*hfc.fabric_network.gateway.Gateway method*), 99
 get_options () (*hfc.fabric_network.contract.Contract method*), 97
 get_options () (*hfc.fabric_network.gateway.Gateway method*), 99
 get_orderer () (*hfc.fabric.Client method*), 77
 get_orderer () (*hfc.fabric.client.Client method*), 62
 get_outof () (*hfc.util.policies.String2Dict method*), 155
 get_peer () (*hfc.fabric.Client method*), 77
 get_peer () (*hfc.fabric.client.Client method*), 62
 get_policy () (*hfc.util.policies.Dict2String method*), 155
 get_public_key () (*hfc.util.crypto.crypto.AsymmetricKey method*), 149
 get_SKI () (*hfc.util.crypto.crypto.Key method*), 149
 get_user () (*hfc.fabric.Client method*), 77
 get_user () (*hfc.fabric.client.Client method*), 62
 get_user () (*hfc.fabric.organization.Organization method*), 73
 get_value () (*hfc.util.keyvaluestore.FileKeyValueStore method*), 154
 get_value () (*hfc.util.keyvaluestore.KeyValueStore method*), 153
 getAll () (*hfc.fabric_ca.affiliationService.AffiliationService method*), 87
 getAll () (*hfc.fabric_ca.identityService.IdentityService method*), 94
 getCertificates () (*hfc.fabric_ca.certificateService.CertificateService method*), 93
 getChannelEventHubsForOrg () (*hfc.fabric.channel.channel.Channel method*), 41
 GetHistoryForKey (*in module hfc.protos.peer.chaincode_shim_pb2*), 132
 GetLogSpec () (*hfc.protos.peer.admin_pb2_grpc.AdminServicer method*), 130
 GetModuleLogLevel () (*hfc.protos.peer.admin_pb2_grpc.AdminServicer method*), 130
 getOne () (*hfc.fabric_ca.affiliationService.AffiliationService method*), 87
 getOne () (*hfc.fabric_ca.identityService.IdentityService method*), 94
 GetQueryResult (*in module hfc.protos.peer.chaincode_shim_pb2*), 132
 GetState (*in module hfc.protos.peer.chaincode_shim_pb2*), 132
 GetStateByRange (*in module hfc.protos.peer.chaincode_shim_pb2*), 132
 GetStateMetadata (*in module hfc.protos.peer.chaincode_shim_pb2*), 132
 GetStatus () (*hfc.protos.peer.admin_pb2_grpc.AdminServicer method*), 129
 GossipHello (*in module hfc.protos.gossip.message_pb2*), 115
 GossipMessage (*in module hfc.protos.gossip.message_pb2*), 115
 GossipServicer (*class in hfc.protos.gossip.message_pb2_grpc*), 116
 GossipStream () (*hfc.protos.gossip.message_pb2_grpc.GossipServicer method*), 116
 GossipStub (*class in*

- hfc.protos.gossip.message_pb2_grpc*), 116
- ## H
- `handle()` (*hfc.fabric.channel.instantiation.Instantiation* method), 44
- `handle()` (*hfc.fabric.channel.invocation.Invocation* method), 46
- `handle_filtered_chaincode()` (*hfc.fabric.channel.channel_eventhub.ChannelEventHub* method), 43
- `handle_filtered_tx()` (*hfc.fabric.channel.channel_eventhub.ChannelEventHub* method), 43
- `handle_full_chaincode()` (*hfc.fabric.channel.channel_eventhub.ChannelEventHub* method), 43
- `handle_full_tx()` (*hfc.fabric.channel.channel_eventhub.ChannelEventHub* method), 43
- `handle_stream()` (*hfc.fabric.channel.channel_eventhub.ChannelEventHub* method), 43
- `has_options` (in module *hfc.protos.ledger.queryresult.kv_query_result_pb2*), 118
- `has_options` (in module *hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2*), 120
- `has_options` (in module *hfc.protos.ledger.rwset.rwset_pb2*), 121
- `hash()` (*hfc.util.crypto.crypto.Ecies* property), 150
- `HashedRWSet` (in module *hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2*), 120
- `HashingAlgorithm` (in module *hfc.protos.common.configuration_pb2*), 106
- `have_registrations()` (*hfc.fabric.channel.channel_eventhub.ChannelEventHub* method), 43
- `Header` (in module *hfc.protos.common.common_pb2*), 104
- `Header` (in module *hfc.protos.token.prover_pb2*), 144
- `HeaderType` (class in *hfc.fabric.block_decoder*), 54
- `HeaderType` (in module *hfc.protos.common.common_pb2*), 102
- hfc* (module), 36
- hfc.fabric* (module), 36
- hfc.fabric.block_decoder* (module), 52
- hfc.fabric.certificateAuthority* (module), 60
- hfc.fabric.channel* (module), 36
- hfc.fabric.channel.channel* (module), 36
- hfc.fabric.channel.channel_configuration* (module), 42
- hfc.fabric.channel.channel_eventhub* (module), 42
- hfc.fabric.channel.instantiation* (module), 44
- hfc.fabric.channel.invocation* (module), 45
- hfc.fabric.client* (module), 61
- hfc.fabric.config* (module), 48
- hfc.fabric.config.default* (module), 48
- hfc.fabric.msp* (module), 48
- hfc.fabric.orderer* (module), 71
- hfc.fabric.organization* (module), 72
- hfc.fabric.peer* (module), 73
- hfc.fabric.transaction* (module), 48
- hfc.fabric.transaction.tx_context* (module), 48
- hfc.fabric.transaction.tx_proposal_request* (module), 49
- hfc.fabric.user* (module), 75
- hfc.fabric_ca* (module), 86
- hfc.fabric_ca.affiliationService* (module), 93
- hfc.fabric_ca.caservice* (module), 88
- hfc.fabric_ca.certificateService* (module), 93
- hfc.fabric_ca.identityService* (module), 94
- hfc.fabric_network* (module), 96
- hfc.fabric_network.contract* (module), 96
- hfc.fabric_network.couchdbwalletstore* (module), 97
- hfc.fabric_network.gateway* (module), 98
- hfc.fabric_network.inmemorywalletstore* (module), 99
- hfc.fabric_network.network* (module), 100
- hfc.fabric_network.wallet* (module), 100
- hfc.protos* (module), 101
- hfc.protos.common* (module), 101
- hfc.protos.common.collection_pb2* (module), 101
- hfc.protos.common.collection_pb2_grpc* (module), 102
- hfc.protos.common.common_pb2* (module), 102
- hfc.protos.common.common_pb2_grpc* (module), 104
- hfc.protos.common.configtx_pb2* (module), 104
- hfc.protos.common.configtx_pb2_grpc* (module), 106
- hfc.protos.common.configuration_pb2* (module), 106
- hfc.protos.common.configuration_pb2_grpc* (module), 107
- hfc.protos.common.ledger_pb2* (module), 107
- hfc.protos.common.ledger_pb2_grpc* (module), 107
- hfc.protos.common.policies_pb2* (module), 107

hfc.protos.common.policies_pb2_grpc (module), 108
 hfc.protos.discovery (module), 108
 hfc.protos.discovery.protocol_pb2 (module), 108
 hfc.protos.discovery.protocol_pb2_grpc (module), 111
 hfc.protos.gossip (module), 112
 hfc.protos.gossip.message_pb2 (module), 112
 hfc.protos.gossip.message_pb2_grpc (module), 116
 hfc.protos.idemix (module), 117
 hfc.protos.idemix.idemix_pb2 (module), 117
 hfc.protos.idemix.idemix_pb2_grpc (module), 118
 hfc.protos.ledger (module), 118
 hfc.protos.ledger.queryresult (module), 118
 hfc.protos.ledger.queryresult.kv_query_result_pb2 (module), 118
 hfc.protos.ledger.queryresult.kv_query_result_pb2_grpc (module), 119
 hfc.protos.ledger.rwset (module), 119
 hfc.protos.ledger.rwset.kvrwset (module), 119
 hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2 (module), 119
 hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2_grpc (module), 120
 hfc.protos.ledger.rwset.rwset_pb2 (module), 120
 hfc.protos.ledger.rwset.rwset_pb2_grpc (module), 121
 hfc.protos.msp (module), 121
 hfc.protos.msp.identities_pb2 (module), 121
 hfc.protos.msp.identities_pb2_grpc (module), 121
 hfc.protos.msp.msp_config_pb2 (module), 121
 hfc.protos.msp.msp_config_pb2_grpc (module), 122
 hfc.protos.msp.msp_principal_pb2 (module), 122
 hfc.protos.msp.msp_principal_pb2_grpc (module), 123
 hfc.protos.orderer (module), 123
 hfc.protos.orderer.ab_pb2 (module), 123
 hfc.protos.orderer.ab_pb2_grpc (module), 124
 hfc.protos.orderer.cluster_pb2 (module), 125
 hfc.protos.orderer.cluster_pb2_grpc (module), 126
 hfc.protos.orderer.configuration_pb2 (module), 126
 hfc.protos.orderer.configuration_pb2_grpc (module), 127
 hfc.protos.orderer.kafka_pb2 (module), 127
 hfc.protos.orderer.kafka_pb2_grpc (module), 128
 hfc.protos.peer (module), 128
 hfc.protos.peer.admin_pb2 (module), 128
 hfc.protos.peer.admin_pb2_grpc (module), 129
 hfc.protos.peer.chaincode_event_pb2 (module), 130
 hfc.protos.peer.chaincode_event_pb2_grpc (module), 130
 hfc.protos.peer.chaincode_pb2 (module), 130
 hfc.protos.peer.chaincode_pb2_grpc (module), 131
 hfc.protos.peer.chaincode_shim_pb2 (module), 131
 hfc.protos.peer.chaincode_shim_pb2_grpc (module), 133
 hfc.protos.peer.configuration_pb2 (module), 134
 hfc.protos.peer.configuration_pb2_grpc (module), 134
 hfc.protos.peer.events_pb2 (module), 134
 hfc.protos.peer.events_pb2_grpc (module), 135
 hfc.protos.peer.lifecycle (module), 128
 hfc.protos.peer.lifecycle.lifecycle_pb2 (module), 128
 hfc.protos.peer.lifecycle.lifecycle_pb2_grpc (module), 128
 hfc.protos.peer.peer_pb2 (module), 136
 hfc.protos.peer.peer_pb2_grpc (module), 136
 hfc.protos.peer.policy_pb2 (module), 137
 hfc.protos.peer.policy_pb2_grpc (module), 137
 hfc.protos.peer.proposal_pb2 (module), 137
 hfc.protos.peer.proposal_pb2_grpc (module), 138
 hfc.protos.peer.proposal_response_pb2 (module), 138
 hfc.protos.peer.proposal_response_pb2_grpc (module), 138
 hfc.protos.peer.query_pb2 (module), 138
 hfc.protos.peer.query_pb2_grpc (module), 139
 hfc.protos.peer.resources_pb2 (module), 139

<code>hfc.protos.peer.resources_pb2_grpc</code> (<i>module</i>), 139	<code>implicit_metapolicy_rule</code> (<i>in module <code>hfc.fabric.block_decoder</code></i>), 54
<code>hfc.protos.peer.signed_cc_dep_spec_pb2</code> (<i>module</i>), 139	<code>ImplicitMetaPolicy</code> (<i>in module <code>hfc.protos.common.policies_pb2</code></i>), 108
<code>hfc.protos.peer.signed_cc_dep_spec_pb2_grpc</code> (<i>module</i>), 140	<code>ImportRequest</code> (<i>in module <code>hfc.protos.token.prover_pb2</code></i>), 144
<code>hfc.protos.peer.transaction_pb2</code> (<i>module</i>), 140	<code>init_with_bundle()</code> (<i><code>hfc.fabric.certificateAuthority.certificateAuthority</code> method</i>), 61
<code>hfc.protos.peer.transaction_pb2_grpc</code> (<i>module</i>), 141	<code>init_with_bundle()</code> (<i><code>hfc.fabric.orderer.Orderer</code> method</i>), 71
<code>hfc.protos.token</code> (<i>module</i>), 141	<code>init_with_bundle()</code> (<i><code>hfc.fabric.organization.Organization</code> method</i>), 72
<code>hfc.protos.token.expectations_pb2</code> (<i>module</i>), 141	<code>init_with_bundle()</code> (<i><code>hfc.fabric.peer.Peer</code> method</i>), 74
<code>hfc.protos.token.expectations_pb2_grpc</code> (<i>module</i>), 142	<code>init_with_discovery()</code> (<i><code>hfc.fabric.Client</code> method</i>), 77
<code>hfc.protos.token.prover_pb2</code> (<i>module</i>), 142	<code>init_with_discovery()</code> (<i><code>hfc.fabric.client.Client</code> method</i>), 61
<code>hfc.protos.token.prover_pb2_grpc</code> (<i>module</i>), 144	<code>init_with_net_profile()</code> (<i><code>hfc.fabric.Client</code> method</i>), 77
<code>hfc.protos.token.transaction_pb2</code> (<i>module</i>), 145	<code>init_with_net_profile()</code> (<i><code>hfc.fabric.client.Client</code> method</i>), 61
<code>hfc.protos.token.transaction_pb2_grpc</code> (<i>module</i>), 146	<code>InMemoryWalletStore</code> (<i>class in <code>hfc.fabric_network.inmemorywalletstore</code></i>), 99
<code>hfc.protos.transientstore</code> (<i>module</i>), 146	<code>InputId</code> (<i>in module <code>hfc.protos.token.transaction_pb2</code></i>), 146
<code>hfc.protos.transientstore.transientstore_pb2</code> (<i>module</i>), 146	<code>InstallChaincodeArgs</code> (<i>in module <code>hfc.protos.peer.lifecycle.lifecycle_pb2</code></i>), 128
<code>hfc.protos.transientstore.transientstore_pb2_grpc</code> (<i>module</i>), 147	<code>InstallChaincodeResult</code> (<i>in module <code>hfc.protos.peer.lifecycle.lifecycle_pb2</code></i>), 128
<code>hfc.protos.utils</code> (<i>module</i>), 147	<code>Instantiation</code> (<i>class in <code>hfc.fabric.channel.instantiation</code></i>), 44
<code>hfc.util</code> (<i>module</i>), 148	<code>INTERNAL_SERVER_ERROR</code> (<i>in module <code>hfc.protos.common.common_pb2</code></i>), 103
<code>hfc.util.channel</code> (<i>module</i>), 152	<code>INVALID_CONFIG_TRANSACTION</code> (<i>in module <code>hfc.protos.peer.transaction_pb2</code></i>), 140
<code>hfc.util.crypto</code> (<i>module</i>), 148	<code>INVALID_ENDORSER_TRANSACTION</code> (<i>in module <code>hfc.protos.peer.transaction_pb2</code></i>), 140
<code>hfc.util.crypto.crypto</code> (<i>module</i>), 148	<code>INVALID_OTHER_REASON</code> (<i>in module <code>hfc.protos.peer.transaction_pb2</code></i>), 141
<code>hfc.util.keyvaluestore</code> (<i>module</i>), 153	<code>INVALID_WRITESET</code> (<i>in module <code>hfc.protos.peer.transaction_pb2</code></i>), 141
<code>hfc.util.policies</code> (<i>module</i>), 155	<code>Invocation</code> (<i>class in <code>hfc.fabric.channel.invocation</code></i>), 46
<code>hfc.util.utils</code> (<i>module</i>), 156	<code>is_dev_mode()</code> (<i><code>hfc.fabric.channel.channel.Channel</code> property</i>), 37
<code>hfc.version</code> (<i>module</i>), 160	<code>is_enrolled()</code> (<i><code>hfc.fabric.user.User</code> method</i>), 76
<code>HMAC_KEY_LENGTH</code> (<i>in module <code>hfc.util.crypto.crypto</code></i>), 148	<code>is_private()</code> (<i><code>hfc.util.crypto.crypto.AsymmetricKey</code> method</i>), 149
<code> </code>	<code>is_readonly()</code> (<i><code>hfc.fabric.channel.channel.Channel</code> method</i>), 38
<code>IdemixMSPConfig</code> (<i>in module <code>hfc.protos.msp.msp_config_pb2</code></i>), 122	
<code>IdemixMSPSignerConfig</code> (<i>in module <code>hfc.protos.msp.msp_config_pb2</code></i>), 122	
<code>Identity</code> (<i>class in <code>hfc.fabric_network.wallet</code></i>), 101	
<code>identity()</code> (<i><code>hfc.fabric.transaction.tx_context.TXContext</code> property</i>), 48	
<code>IDENTITY_MSG</code> (<i>in module <code>hfc.protos.gossip.message_pb2</code></i>), 112	
<code>IdentityService</code> (<i>class in <code>hfc.fabric_ca.identityService</code></i>), 94	
<code>ILLEGAL_WRITESET</code> (<i>in module <code>hfc.protos.peer.transaction_pb2</code></i>), 141	

- is_registered() (*hfc.fabric.user.User* method), 75
- is_symmetric() (*hfc.util.crypto.crypto.Key* method), 148
- is_sys_chan() (*hfc.fabric.channel.channel.Channel* property), 37
- IssuerKey (in module *hfc.protos.idemix.idemix_pb2*), 118
- IssuerPublicKey (in module *hfc.protos.idemix.idemix_pb2*), 118
- IV_LENGTH (in module *hfc.util.crypto.crypto*), 148
- ## J
- join() (*hfc.fabric.peer.Peer* method), 74
- join_channel() (*hfc.fabric.channel.channel.Channel* method), 38
- ## K
- KafkaBrokers (in module *hfc.protos.orderer.configuration_pb2*), 127
- KafkaMessage (in module *hfc.protos.orderer.kafka_pb2*), 127
- KafkaMessageConnect (in module *hfc.protos.orderer.kafka_pb2*), 127
- KafkaMessageRegular (in module *hfc.protos.orderer.kafka_pb2*), 127
- KafkaMessageTimeToCut (in module *hfc.protos.orderer.kafka_pb2*), 127
- KafkaMetadata (in module *hfc.protos.orderer.kafka_pb2*), 127
- Key (class in *hfc.util.crypto.crypto*), 148
- KeyInfo (in module *hfc.protos.msp.msp_config_pb2*), 122
- KeyModification (in module *hfc.protos.ledger.queryresult.kv_query_result_pb2*), 118
- KeyValueStore (class in *hfc.util.keyvaluestore*), 153
- KV (in module *hfc.protos.ledger.queryresult.kv_query_result_pb2*), 118
- KVRead (in module *hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2*), 120
- KVReadHash (in module *hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2*), 120
- KVRWSet (in module *hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2*), 119
- KVWrite (in module *hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2*), 120
- KVWriteHash (in module *hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2*), 120
- ## L
- LAST_CONFIG (in module *hfc.protos.common.common_pb2*), 103
- LastConfig (in module *hfc.protos.common.common_pb2*), 103
- Layout (in module *hfc.protos.discovery.protocol_pb2*), 110
- LeadershipMessage (in module *hfc.protos.gossip.message_pb2*), 115
- LifecycleEvent (in module *hfc.protos.peer.chaincode_pb2*), 131
- ListRequest (in module *hfc.protos.token.prover_pb2*), 144
- LocalPeerQuery (in module *hfc.protos.discovery.protocol_pb2*), 110
- logic() (*hfc.util.policies.String2Dict* method), 155
- LogLevelRequest (in module *hfc.protos.peer.admin_pb2*), 129
- LogLevelResponse (in module *hfc.protos.peer.admin_pb2*), 129
- LogSpecRequest (in module *hfc.protos.peer.admin_pb2*), 129
- LogSpecResponse (in module *hfc.protos.peer.admin_pb2*), 129
- ## M
- MARSHAL_TX_ERROR (in module *hfc.protos.peer.transaction_pb2*), 140
- Member (in module *hfc.protos.gossip.message_pb2*), 115
- MembershipRequest (in module *hfc.protos.gossip.message_pb2*), 115
- MembershipResponse (in module *hfc.protos.gossip.message_pb2*), 115
- mergeOptions() (*hfc.fabric_network.gateway.Gateway* method), 98
- MESSAGE (in module *hfc.protos.common.common_pb2*), message_type (in module *hfc.protos.common.collection_pb2*), 102
- message_type (in module *hfc.protos.common.common_pb2*), 103
- message_type (in module *hfc.protos.common.configtx_pb2*), 105
- message_type (in module *hfc.protos.common.configuration_pb2*), 106
- message_type (in module *hfc.protos.common.policies_pb2*), 107
- message_type (in module *hfc.protos.discovery.protocol_pb2*), 109, 110
- message_type (in module *hfc.protos.gossip.message_pb2*), 113–115
- message_type (in module *hfc.protos.idemix.idemix_pb2*), 117

message_type	(in module	147	
	<i>hfc.protos.ledger.queryresult.kv_query_result_pb2</i>),		Metadata (in module
118			<i>hfc.protos.common.common_pb2</i>), 103
message_type	(in module		MetadataKeys (in module
	<i>hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2</i>),		<i>hfc.protos.peer.transaction_pb2</i>), 140
119			MetadataSignature (in module
message_type	(in module		<i>hfc.protos.common.common_pb2</i>), 104
	<i>hfc.protos.ledger.rwset.rwset_pb2</i>), 120		msp_id() (<i>hfc.fabric.user.User</i> property), 75
message_type	(in module		MSPConfig (in module
	<i>hfc.protos.msp.msp_config_pb2</i>), 122		<i>hfc.protos.msp.msp_config_pb2</i>), 122
message_type	(in module		msp_id() (<i>hfc.util.policies.String2Dict</i> method), 155
	<i>hfc.protos.msp.msp_principal_pb2</i>), 123		MSPIdentityAnonymity (in module
message_type (in module <i>hfc.protos.orderer.ab_pb2</i>),			<i>hfc.protos.msp.msp_principal_pb2</i>), 123
124			MSPPrincipal (in module
message_type	(in module		<i>hfc.protos.msp.msp_principal_pb2</i>), 123
	<i>hfc.protos.orderer.cluster_pb2</i>), 125		MSPRole (in module
message_type	(in module		<i>hfc.protos.msp.msp_principal_pb2</i>), 123
	<i>hfc.protos.orderer.kafka_pb2</i>), 127		MVCC_READ_CONFLICT (in module
message_type	(in module		<i>hfc.protos.peer.transaction_pb2</i>), 140
	<i>hfc.protos.peer.admin_pb2</i>), 129		
message_type	(in module		N
	<i>hfc.protos.peer.chaincode_pb2</i>), 131		name() (<i>hfc.fabric.channel.channel.Channel</i> property),
message_type	(in module		37
	<i>hfc.protos.peer.chaincode_shim_pb2</i>), 132		name() (<i>hfc.fabric.orderer.Orderer</i> property), 72
message_type	(in module		name() (<i>hfc.fabric.peer.Peer</i> property), 74
	<i>hfc.protos.peer.configuration_pb2</i>), 134		name() (<i>hfc.fabric.user.User</i> property), 75
message_type	(in module		name() (<i>hfc.util.policies.String2Dict</i> method), 155
	<i>hfc.protos.peer.events_pb2</i>), 134, 135		Network (class in <i>hfc.fabric_network.network</i>), 100
message_type (in module <i>hfc.protos.peer.peer_pb2</i>),			new_channel() (<i>hfc.fabric.Client</i> method), 78
136			new_channel() (<i>hfc.fabric.client.Client</i> method), 62
message_type	(in module		newAffiliationService()
	<i>hfc.protos.peer.policy_pb2</i>), 137		(<i>hfc.fabric_ca.CAClient</i> method), 96
message_type	(in module		newAffiliationService()
	<i>hfc.protos.peer.proposal_pb2</i>), 137		(<i>hfc.fabric_ca.caservice.CAClient</i> method),
message_type	(in module		90
	<i>hfc.protos.peer.proposal_response_pb2</i>),		newAffiliationService()
138			(<i>hfc.fabric_ca.caservice.CAService</i> method),
message_type (in module <i>hfc.protos.peer.query_pb2</i>),			92
138, 139			newCertificateService()
message_type	(in module		(<i>hfc.fabric_ca.CAClient</i> method), 96
	<i>hfc.protos.peer.resources_pb2</i>), 139		newCertificateService()
message_type	(in module		(<i>hfc.fabric_ca.caservice.CAClient</i> method),
	<i>hfc.protos.peer.signed_cc_dep_spec_pb2</i>),		90
140			newCertificateService()
message_type	(in module		(<i>hfc.fabric_ca.caservice.CAService</i> method),
	<i>hfc.protos.peer.transaction_pb2</i>), 141		92
message_type	(in module		newChannelEventHub()
	<i>hfc.protos.token.expectations_pb2</i>), 142		(<i>hfc.fabric.channel.channel.Channel</i> method),
message_type	(in module		41
	<i>hfc.protos.token.prover_pb2</i>), 143		newIdentityService() (<i>hfc.fabric_ca.CAClient</i>
message_type	(in module		method), 96
	<i>hfc.protos.token.transaction_pb2</i>), 145, 146		newIdentityService()
message_type	(in module		(<i>hfc.fabric_ca.caservice.CAClient</i> method),
	<i>hfc.protos.transientstore.transientstore_pb2</i>),		90

newIdentityService() (*hfc.fabric_ca.caservice.CAService* method), 92
 NIL_ENVELOPE (in module *hfc.protos.peer.transaction_pb2*), 140
 NIL_TXACTION (in module *hfc.protos.peer.transaction_pb2*), 140
 NO_START_STOP (in module *hfc.fabric.channel.channel_eventhub*), 42
 nonce() (*hfc.fabric.transaction.tx_context.TXContext* property), 48
 NonRevocationProof (in module *hfc.protos.idemix.idemix_pb2*), 118
 NOT_FOUND (in module *hfc.protos.common.common_pb2*), 102
 NOT_IMPLEMENTED (in module *hfc.protos.common.common_pb2*), 103
 NOT_VALIDATED (in module *hfc.protos.peer.transaction_pb2*), 141
 NsPvtReadWriteSet (in module *hfc.protos.ledger.rwset.rwset_pb2*), 121
 NsReadWriteSet (in module *hfc.protos.ledger.rwset.rwset_pb2*), 121
 NullHandler (class in *hfc.fabric*), 86
 NullHandler (class in *hfc.fabric.channel*), 47
 NullHandler (class in *hfc.fabric.transaction*), 52
 NullHandler (class in *hfc.fabric_ca*), 96
 NullHandler (class in *hfc.util*), 159
 NullHandler (class in *hfc.util.crypto*), 152
 number() (*hfc.util.policies.String2Dict* method), 155
 NymSignature (in module *hfc.protos.idemix.idemix_pb2*), 118

O

Orderer (class in *hfc.fabric.orderer*), 71
 ORDERER (in module *hfc.protos.common.common_pb2*), 103
 ORDERER_TRANSACTION (in module *hfc.protos.common.common_pb2*), 103
 OrdererAddresses (in module *hfc.protos.common.configuration_pb2*), 106
 orderers() (*hfc.fabric.channel.channel.Channel* property), 37
 orderers() (*hfc.fabric.Client* property), 77
 orderers() (*hfc.fabric.client.Client* property), 62
 org() (*hfc.fabric.user.User* property), 75
 Organization (class in *hfc.fabric.organization*), 72
 organizations() (*hfc.fabric.Client* property), 77
 organizations() (*hfc.fabric.client.Client* property), 62
 OrganizationUnit (in module *hfc.protos.msp.msp_principal_pb2*), 123

P

package_chaincode() (in module *hfc.util.utils*), 159
 packaged_cc() (*hfc.fabric.transaction.tx_proposal_request.TXProposal* property), 51
 parse() (*hfc.util.policies.Dict2String* method), 155
 Payload (in module *hfc.protos.common.common_pb2*), 104
 Payload (in module *hfc.protos.gossip.message_pb2*), 115
 Peer (class in *hfc.fabric.peer*), 73
 Peer (in module *hfc.protos.discovery.protocol_pb2*), 110
 PEER_ADMIN_OPERATION (in module *hfc.protos.common.common_pb2*), 103
 PeerEndpoint (in module *hfc.protos.peer.peer_pb2*), 136
 PeerID (in module *hfc.protos.peer.peer_pb2*), 136
 PeerIdentity (in module *hfc.protos.gossip.message_pb2*), 115
 PeerMembershipQuery (in module *hfc.protos.discovery.protocol_pb2*), 110
 PeerMembershipResult (in module *hfc.protos.discovery.protocol_pb2*), 110
 Peers (in module *hfc.protos.discovery.protocol_pb2*), 110
 peers() (*hfc.fabric.channel.channel.Channel* property), 37
 peers() (*hfc.fabric.Client* property), 78
 peers() (*hfc.fabric.client.Client* property), 62
 PeerTime (in module *hfc.protos.gossip.message_pb2*), 115
 pem_to_der() (in module *hfc.util.utils*), 159
 PHANTOM_READ_CONFLICT (in module *hfc.protos.peer.transaction_pb2*), 140
 Ping() (*hfc.protos.gossip.message_pb2_grpc.GossipServicer* method), 116
 PlainApprove (in module *hfc.protos.token.transaction_pb2*), 146
 PlainDelegatedOutput (in module *hfc.protos.token.transaction_pb2*), 146
 PlainExpectation (in module *hfc.protos.token.expectations_pb2*), 142
 PlainImport (in module *hfc.protos.token.transaction_pb2*), 146
 PlainOutput (in module *hfc.protos.token.transaction_pb2*), 146
 PlainTokenAction (in module *hfc.protos.token.transaction_pb2*), 146
 PlainTokenExpectation (in module *hfc.protos.token.expectations_pb2*), 142
 PlainTransfer (in module *hfc.protos.token.transaction_pb2*), 146
 PlainTransferFrom (in module *hfc.protos.token.transaction_pb2*), 146

Policy (in module *hfc.protos.common.policies_pb2*), 108
 policy_policy_type (in module *hfc.fabric.block_decoder*), 54
 private_key() (*hfc.fabric_ca.caservice.EnrollmentProperty*), 89
 PrivateDataMessage (in module *hfc.protos.gossip.message_pb2*), 115
 PrivatePayload (in module *hfc.protos.gossip.message_pb2*), 115
 ProcessCommand() (*hfc.protos.token.prover_pb2_grpc.ProverServicer* method), 145
 ProcessedTransaction (in module *hfc.protos.peer.transaction_pb2*), 141
 ProcessProposal() (*hfc.protos.peer.peer_pb2_grpc.EndorserServicer* method), 136
 prop_type() (*hfc.fabric.transaction.tx_proposal_request.TXProposalRequest* property), 50
 prop_wait_time() (*hfc.fabric.transaction.tx_context.TXContext* property), 49
 Properties (in module *hfc.protos.gossip.message_pb2*), 115
 Proposal (in module *hfc.protos.peer.proposal_pb2*), 137
 ProposalResponse (in module *hfc.protos.peer.proposal_response_pb2*), 138
 ProposalResponsePayload (in module *hfc.protos.peer.proposal_response_pb2*), 138
 proto_b (in module *hfc.util.utils*), 156
 proto_str() (in module *hfc.util.utils*), 156
 ProverServicer (class in *hfc.protos.token.prover_pb2_grpc*), 145
 ProverStub (class in *hfc.protos.token.prover_pb2_grpc*), 145
 PUBLIC (in module *hfc.protos.peer.chaincode_pb2*), 130
 PullMsgType (in module *hfc.protos.gossip.message_pb2*), 112
 put() (*hfc.fabric_network.couchdbwalletstore.CouchDBWalletStore* method), 98
 put() (*hfc.fabric_network.inmemorywalletstore.InMemoryWalletStore* method), 99
 PutState (in module *hfc.protos.peer.chaincode_shim_pb2*), 132
 PutStateMetadata (in module *hfc.protos.peer.chaincode_shim_pb2*), 132
 PvtDataDigest (in module *hfc.protos.gossip.message_pb2*), 116
 PvtDataElement (in module *hfc.protos.gossip.message_pb2*), 116
 PvtDataPayload (in module *hfc.protos.gossip.message_pb2*), 116
 Query (in module *hfc.protos.discovery.protocol_pb2*), 110
 query_block() (*hfc.fabric.channel.channel.Channel* method), 40
 query_block() (*hfc.fabric.Client* method), 84
 query_block() (*hfc.fabric.client.Client* method), 69
 query_block_by_hash() (*hfc.fabric.channel.channel.Channel* method), 40
 query_block_by_hash() (*hfc.fabric.Client* method), 84
 query_block_by_hash() (*hfc.fabric.client.Client* method), 69
 query_block_by_txid() (*hfc.fabric.channel.channel.Channel* method), 40
 query_block_by_txid() (*hfc.fabric.Client* method), 84
 query_block_by_txid() (*hfc.fabric.client.Client* method), 69
 query_channels() (*hfc.fabric.Client* method), 84
 query_channels() (*hfc.fabric.client.Client* method), 68
 query_info() (*hfc.fabric.channel.channel.Channel* method), 40
 query_info() (*hfc.fabric.Client* method), 84
 query_info() (*hfc.fabric.client.Client* method), 68
 query_installed_chaincodes() (*hfc.fabric.Client* method), 85
 query_installed_chaincodes() (*hfc.fabric.client.Client* method), 70
 query_instantiated_chaincodes() (*hfc.fabric.channel.channel.Channel* method), 39
 query_instantiated_chaincodes() (*hfc.fabric.Client* method), 85
 query_instantiated_chaincodes() (*hfc.fabric.client.Client* method), 70
 query_peers() (*hfc.fabric.Client* method), 85
 query_peers() (*hfc.fabric.client.Client* method), 70
 query_transaction() (*hfc.fabric.channel.channel.Channel* method), 39
 query_transaction() (*hfc.fabric.Client* method), 85
 query_transaction() (*hfc.fabric.client.Client* method), 69
 QueryInstalledChaincodeArgs (in module *hfc.protos.peer.lifecycle.lifecycle_pb2*), 128
 QueryInstalledChaincodeResult (in module *hfc.protos.peer.lifecycle.lifecycle_pb2*), 128
 QueryMetadata (in module *hfc.protos.peer.chaincode_shim_pb2*), 132

QueryReads (in module *hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2*), 120
 QueryReadsMerkleSummary (in module *hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2*), 120
 QueryResponse (in module *hfc.protos.peer.chaincode_shim_pb2*), 133
 QueryResponseMetadata (in module *hfc.protos.peer.chaincode_shim_pb2*), 133
 QueryResult (in module *hfc.protos.discovery.protocol_pb2*), 110
 QueryResultBytes (in module *hfc.protos.peer.chaincode_shim_pb2*), 133
 QueryStateClose (in module *hfc.protos.peer.chaincode_shim_pb2*), 132
 QueryStateNext (in module *hfc.protos.peer.chaincode_shim_pb2*), 132

R

RangeQueryInfo (in module *hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2*), 120
 reasons (in module *hfc.fabric_ca.caservice*), 88
 RecipientTransferShare (in module *hfc.protos.token.prover_pb2*), 144
 RedeemRequest (in module *hfc.protos.token.prover_pb2*), 144
 reenroll() (*hfc.fabric_ca.CAClient* method), 96
 reenroll() (*hfc.fabric_ca.caservice.CAClient* method), 90
 reenroll() (*hfc.fabric_ca.caservice.CAService* method), 91
 register() (*hfc.fabric_ca.CAClient* method), 96
 register() (*hfc.fabric_ca.caservice.CAClient* method), 90
 register() (*hfc.fabric_ca.caservice.CAService* method), 91
 register() (*hfc.fabric_ca.caservice.Enrollment* method), 89
 Register() (*hfc.protos.peer.chaincode_shim_pb2_grpc.ChaincodeSupportService* method), 133
 registerBlockEvent() (*hfc.fabric.channel.channel_eventhub.ChannelEventHub* method), 43
 registerChaincodeEvent() (*hfc.fabric.channel.channel_eventhub.ChannelEventHub* method), 43
 registerTxEvent() (*hfc.fabric.channel.channel_eventhub.ChannelEventHub* method), 43
 RemotePvtDataRequest (in module *hfc.protos.gossip.message_pb2*), 115
 RemotePvtDataResponse (in module *hfc.protos.gossip.message_pb2*), 116
 RemoteStateRequest (in module *hfc.protos.gossip.message_pb2*), 115
 RemoteStateResponse (in module *hfc.protos.gossip.message_pb2*), 115
 remove() (*hfc.fabric_network.couchdbwalletstore.CouchDBWalletStore* method), 97
 remove() (*hfc.fabric_network.inmemorywalletstore.InMemoryWalletStore* method), 99
 remove() (*hfc.fabric_network.wallet.FileSystemWallet* method), 101
 remove_orderer() (*hfc.fabric.channel.channel.Channel* method), 36
 remove_peer() (*hfc.fabric.channel.channel.Channel* method), 37
 Request (in module *hfc.protos.discovery.protocol_pb2*), 110
 REQUEST_ENTITY_TOO_LARGE (in module *hfc.protos.common.common_pb2*), 102
 Response (in module *hfc.protos.discovery.protocol_pb2*), 110
 Response (in module *hfc.protos.peer.proposal_response_pb2*), 138
 RevertLogLevels() (*hfc.protos.peer.admin_pb2_grpc.AdminService* method), 130
 revoke() (*hfc.fabric_ca.CAClient* method), 96
 revoke() (*hfc.fabric_ca.caservice.CAClient* method), 90
 revoke() (*hfc.fabric_ca.caservice.CAService* method), 92
 revoke() (*hfc.fabric_ca.caservice.Enrollment* method), 89
 role() (*hfc.util.policies.String2Dict* method), 155
 roles (*hfc.util.policies.Dict2String* attribute), 155
 roles() (*hfc.fabric.user.User* property), 75

S

s2d_grammar (in module *hfc.util.policies*), 155
 Secret (in module *hfc.protos.gossip.message_pb2*), 115
 SecretEnvelope (in module *hfc.protos.gossip.message_pb2*), 115
 SeekInfo (in module *hfc.protos.orderer.ab_pb2*), 124
 SeekNewest (in module *hfc.protos.orderer.ab_pb2*), 124
 SeekOldest (in module *hfc.protos.orderer.ab_pb2*), 124
 SeekPosition (in module *hfc.protos.orderer.ab_pb2*), 124
 SeekSpecified (in module *hfc.protos.orderer.ab_pb2*), 124

send_discovery() (*hfc.fabric.peer.Peer* method), 73
 send_install_proposal() (*hfc.fabric.channel.channel.Channel* method), 38
 send_install_proposal() (*hfc.fabric.Client* method), 80
 send_install_proposal() (*hfc.fabric.client.Client* method), 65
 send_install_proposal() (in module *hfc.util.utils*), 159
 send_instantiate_proposal() (*hfc.fabric.channel.channel.Channel* method), 38
 send_instantiate_proposal() (*hfc.fabric.Client* method), 80
 send_instantiate_proposal() (*hfc.fabric.client.Client* method), 65
 send_proposal() (*hfc.fabric.peer.Peer* method), 73
 send_transaction() (in module *hfc.util.utils*), 158
 send_transaction_proposal() (in module *hfc.util.utils*), 158
 send_tx_proposal() (*hfc.fabric.channel.channel.Channel* method), 39
 send_upgrade_proposal() (*hfc.fabric.channel.channel.Channel* method), 38
 send_upgrade_proposal() (*hfc.fabric.Client* method), 80
 send_upgrade_proposal() (*hfc.fabric.client.Client* method), 65
 SerializedIdemixIdentity (in module *hfc.protos.msp.identities_pb2*), 121
 SerializedIdentity (in module *hfc.protos.msp.identities_pb2*), 121
 ServerStatus (in module *hfc.protos.peer.admin_pb2*), 129
 SERVICE_UNAVAILABLE (in module *hfc.protos.common.common_pb2*), 103
 set_tls_client_cert_and_key() (*hfc.fabric.Client* method), 77
 set_tls_client_cert_and_key() (*hfc.fabric.client.Client* method), 62
 set_tls_client_cert_and_key() (*hfc.fabric.orderer.Orderer* method), 72
 set_tls_client_cert_and_key() (*hfc.fabric.peer.Peer* method), 74
 set_value() (*hfc.util.keyvaluestore.FileKeyValueStore* method), 154
 set_value() (*hfc.util.keyvaluestore.KeyValueStore* method), 153
 SetLogSpec() (*hfc.protos.peer.admin_pb2_grpc.AdminService* method), 130
 SetModuleLogLevel() (*hfc.protos.peer.admin_pb2_grpc.AdminService* method), 130
 SHA2 (in module *hfc.util.crypto.crypto*), 148
 SHA3 (in module *hfc.util.crypto.crypto*), 148
 sign() (*hfc.fabric.transaction.tx_context.TXContext* method), 49
 sign() (*hfc.util.crypto.crypto.Crypto* method), 149
 sign() (*hfc.util.crypto.crypto.Ecies* method), 150
 sign_channel_config() (*hfc.fabric.Client* method), 79
 sign_channel_config() (*hfc.fabric.client.Client* method), 64
 sign_proposal() (in module *hfc.util.utils*), 158
 sign_tran_payload() (in module *hfc.util.utils*), 158
 Signature (in module *hfc.protos.idemix.idemix_pb2*), 118
 SignatureHeader (in module *hfc.protos.common.common_pb2*), 104
 SignaturePolicy (in module *hfc.protos.common.policies_pb2*), 108
 SignaturePolicyEnvelope (in module *hfc.protos.common.policies_pb2*), 108
 SIGNATURES (in module *hfc.protos.common.common_pb2*), 103
 SignedChaincodeDeploymentSpec (in module *hfc.protos.peer.signed_cc_dep_spec_pb2*), 140
 SignedCommand (in module *hfc.protos.token.prover_pb2*), 144
 SignedCommandResponse (in module *hfc.protos.token.prover_pb2*), 144
 SignedProposal (in module *hfc.protos.peer.proposal_pb2*), 137
 SignedRequest (in module *hfc.protos.discovery.protocol_pb2*), 110
 SignedTransaction (in module *hfc.protos.peer.transaction_pb2*), 141
 SigningIdentityInfo (in module *hfc.protos.msp.msp_config_pb2*), 122
 START_AND_END (in module *hfc.fabric.channel.channel_eventhub*), 42
 START_ONLY (in module *hfc.fabric.channel.channel_eventhub*), 42
 StartServer() (*hfc.protos.peer.admin_pb2_grpc.AdminService* method), 129
 state_store() (*hfc.fabric.channel.channel.Channel* method), 37
 state_store() (*hfc.fabric.Client* property), 80
 state_store() (*hfc.fabric.client.Client* property), 65
 StateInfo (in module *hfc.protos.gossip.message_pb2*), 115
 StateInfoPullRequest (in module *hfc.protos.gossip.message_pb2*), 115
 StateInfoSnapshot (in module

hfc.protos.gossip.message_pb2), 115
 StateMetadata (in module *hfc.protos.peer.chaincode_shim_pb2*), 133
 StateMetadataResult (in module *hfc.protos.peer.chaincode_shim_pb2*), 133
 StaticCollectionConfig (in module *hfc.protos.common.collection_pb2*), 102
 Status (in module *hfc.protos.common.common_pb2*), 102
 Step() (*hfc.protos.orderer.cluster_pb2_grpc.ClusterServicer* method), 126
 StepRequest (in module *hfc.protos.orderer.cluster_pb2*), 125
 StepResponse (in module *hfc.protos.orderer.cluster_pb2*), 125
 stream_envelope() (in module *hfc.util.utils*), 159
 String2Dict (class in *hfc.util.policies*), 155
 string_to_signature() (in module *hfc.util.utils*), 157
 Submit() (*hfc.protos.orderer.cluster_pb2_grpc.ClusterServicer* method), 126
 submit_transaction() (*hfc.fabric_network.contract.Contract* method), 97
 SubmitRequest (in module *hfc.protos.orderer.cluster_pb2*), 125
 SubmitResponse (in module *hfc.protos.orderer.cluster_pb2*), 125
 SUCCESS (in module *hfc.protos.common.common_pb2*), 102
 SYSTEM_CHANNEL_NAME (in module *hfc.fabric.channel.channel*), 36

T

TARGET_CHAIN_NOT_FOUND (in module *hfc.protos.peer.transaction_pb2*), 140
 timestamp_to_date() (in module *hfc.fabric.block_decoder*), 55
 to_PEM_certs() (in module *hfc.fabric.block_decoder*), 58
 TOKEN_TRANSACTION (in module *hfc.protos.common.common_pb2*), 103
 TokenExpectation (in module *hfc.protos.token.expectations_pb2*), 142
 TokenOutput (in module *hfc.protos.token.prover_pb2*), 144
 TokenToIssue (in module *hfc.protos.token.prover_pb2*), 144
 TokenTransaction (in module *hfc.protos.token.transaction_pb2*), 146
 Transaction (in module *hfc.protos.peer.transaction_pb2*), 141
 TransactionAction (in module *hfc.protos.peer.transaction_pb2*), 141
 TRANSACTIONS_FILTER (in module *hfc.protos.common.common_pb2*), 103
 TransferRequest (in module *hfc.protos.token.prover_pb2*), 144
 transient_map() (*hfc.fabric.transaction.tx_proposal_request.TXProposalRequest* property), 51
 tx_context() (*hfc.fabric.Client* property), 80
 tx_context() (*hfc.fabric.client.Client* property), 65
 tx_id() (*hfc.fabric.transaction.tx_context.TXContext* property), 48
 tx_prop_req() (*hfc.fabric.transaction.tx_context.TXContext* property), 49
 tx_validation_code (in module *hfc.fabric.block_decoder*), 54
 TXContext (class in *hfc.fabric.transaction.tx_context*), 48
 txEvent() (*hfc.fabric.Client* method), 81
 txEvent() (*hfc.fabric.client.Client* method), 66
 TXProposalRequest (class in *hfc.fabric.transaction.tx_proposal_request*), 50
 TxPvtReadWriteSet (in module *hfc.protos.ledger.rwset.rwset_pb2*), 121
 TxPvtReadWriteSetWithConfigInfo (in module *hfc.protos.transientstore.transientstore_pb2*), 147
 TxReadWriteSet (in module *hfc.protos.ledger.rwset.rwset_pb2*), 120
 TxValidationCode (in module *hfc.protos.peer.transaction_pb2*), 140
 type_as_string (in module *hfc.fabric.block_decoder*), 54

U

UNDEFINED (in module *hfc.protos.gossip.message_pb2*), 112
 unique_list_of_dict() (*hfc.util.policies.String2Dict* method), 155
 UNKNOWN (in module *hfc.protos.common.common_pb2*), 102
 UNKNOWN_TX_TYPE (in module *hfc.protos.peer.transaction_pb2*), 140
 unregisterBlockEvent() (*hfc.fabric.channel.channel_eventhub.ChannelEventHub* method), 43
 unregisterChaincodeEvent() (*hfc.fabric.channel.channel_eventhub.ChannelEventHub* method), 43
 unregisterTxEvent() (*hfc.fabric.channel.channel_eventhub.ChannelEventHub* method), 43
 UnspentTokens (in module *hfc.protos.token.prover_pb2*), 144
 UNSUPPORTED_TX_PAYLOAD (in module *hfc.protos.peer.transaction_pb2*), 140

`update()` (*hfc.fabric_ca.affiliationService.AffiliationService method*), 88
`update()` (*hfc.fabric_ca.identityService.IdentityService method*), 94
`User` (*class in hfc.fabric.user*), 75
`user()` (*hfc.fabric.transaction.tx_context.TXContext property*), 49

V

`VALID` (*in module hfc.protos.peer.transaction_pb2*), 140
`validate()` (*in module hfc.fabric.transaction.tx_context*), 49
`validate()` (*in module hfc.fabric.transaction.tx_proposal_request*), 51
`validate()` (*in module hfc.fabric.user*), 76
`VALIDATION_PARAMETER` (*in module hfc.protos.peer.transaction_pb2*), 141
`verify()` (*hfc.util.crypto.crypto.Crypto method*), 149
`verify()` (*hfc.util.crypto.crypto.Ecies method*), 150
`VERSION` (*in module hfc*), 160
`Version` (*in module hfc.protos.ledger.rwset.kvrwset.kv_rwset_pb2*), 120
`VERSION` (*in module hfc.version*), 160
`VSCCArgs` (*in module hfc.protos.peer.resources_pb2*), 139

Z

`zeroTarInfo()` (*in module hfc.util.utils*), 159
`zeroTimeContextManager` (*class in hfc.util.utils*), 159