fabric-sdk-py

Release 0.9.0

Hyperledger Community

Jul 09, 2020

CONTENTS

CHAPTER

ONE

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.

CHAPTER

TWO

PRE-REQUISITES

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

CHAPTER

THREE

CREATING A VIRTUAL EVIRONMENT

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  # deactive the virtual env
$ make clean # clean the temporary files
```

CHAPTER

FOUR

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
```

CHAPTER

FIVE

REPORTING BUG

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

CHAPTER

SIX

ASK QUESTIONS

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

CHAPTER

SEVEN

GET STARTED AND READ OUR DOCUMENTATION

You can find our documentation here.

CHAPTER

EIGHT

FEEL FREE TO CONTRIBUTE

Let's get started and contribute to Fabric-SDK-Py! You can start here.

CHAPTER

NINE

SOME IMPORTANT LINKS

CHAPTER

TEN

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.

CHAPTER

ELEVEN

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
$ deactive
```

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 credentail 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 Credentail 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.FileSystenWallet() # 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␣
↪commited 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

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 collcetion 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

../../CONTRIBUTING.md

11.4 API Reference

This page contains auto-generated API reference documentation[1].

[1] Created with sphinx-autoapi

11.4.1 hfc

Subpackages

hfc.fabric

Subpackages

hfc.fabric.channel

Submodules

hfc.fabric.channel.channel

Module Contents

Classes

Channel(name, client)	The class represents the channel.

Functions

create_system_channel(client, name=SYSTEM_CHANNEL_NAME)	Create system channel instance
create_app_channel(client, name)	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*, *msps=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 pees 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

ChannelConfiguration(config=None, file_path=None)	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

EventRegistration(onEvent=None, unregister=True, disconnect=False)	A class represents event registration.
ChaincodeRegistration(ccid, pattern, er, as_array)	A class represents chaincode registration.
ChannelEventHub(peer, channel_name, requestor)	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*, *onEvent=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*, *onEvent=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

Instantiation()	Chaincode instantiate transaction proposal handler.

Functions

_create_instantiation_proposal(tran_prop_req, chain)	Create a chaincode instantiation proposal
_instantiate_chaincode(chain, cc_instantiation_request, scheduler=None)	Instantiate chaincode.
create_instantiation_proposal_req(chaincode_id, chaincode_path, chaincode_version, creator, fcn='init', args=None, nonce=crypto.generate_nonce(24), targets=None)	Create instantiation proposal request.
chaincode_instantiation(chain)	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*, *chaincode_path*, *chaincode_version*, *creator*, *fcn='init'*, *args=None*, *nonce=crypto.generate_nonce(24)*, *targets=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

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

NullHandler(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

validate(tx_context)	Validate transaction context
create_tx_context(user, crypto, tx_prop_req, prop_wait_time=-1)	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

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)	Class represents transaction proposal request.

Functions

validate(tx_prop_req)	Check transaction 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

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

NullHandler(level=NOTSET)	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

BlockDecoder()	An object of a fully decoded protobuf message "Block"
FilteredBlockDecoder()	An object of a fully decoded protobuf message "FilteredBlock"
HeaderType()	HeaderType class having decodePayload and convertToString methods

Functions

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

Continued on next page

Table 16 – continued from previous page

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

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

create_ca(name, info)	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

Client(net_profile=None)	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
- **deocode** – 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 ConfigEnveloppe

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*)
Inits 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
- **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
- **deocode** – 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

Orderer(name='orderer', endpoint=DEFAULT_ORDERER_ENDPOINT, tls_ca_cert_file=None, client_key_file=None, client_cert_file=None, opts=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

Organization(name='org', state_store=None)	An organization in the network.

Functions

create_org(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, tls_ca_cert_file=None, client_key_file=None, client_cert_file=None, opts=None)	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

User(name, org, state_store)	The default implementation of user.

Functions

validate(user)	Check the user.
create_user(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

Client(net_profile=None)	Main interaction handler with end user.
NullHandler(level=NOTSET)	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
- **deocode** – 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 ConfigEnveloppe

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*)
Inits 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
- **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
- **deocode** – 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*)

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

Enrollment(private_key, enrollmentCert, caCertChain=None, service=None)	Class represents enrollment.
CAClient(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.
CAService(target=DEFAULT_CA_ENDPOINT, ca_certs_path=None, crypto=ecies(), ca_name='')	This is a ca server delegate.

Functions

ca_service(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*, *maxEnrollments=1*, *attrs=None*)

revoke(*self*, *enrollmentID=None*, *aki=None*, *serial=None*, *reason=None*, *gencrl=False*)

generateCRL(*self*, *revokedBefore=None*, *revokedAfter=None*, *expireBefore=None*, *expireAfter=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*, *gencrl*, *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
- **gencrl** (*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

CAClient(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.
NullHandler(level=NOTSET)	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=''*, *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.**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

Contract(network, cc_name, gateway)	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 networks'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

CouchDBWalletStore(dbName, config='http://localhost:5984')	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

Network(gateway, channel)	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

FileSystenWallet(path=os.getcwd() + '/tmp/hfc-kvs')	FileSystemWallet stores the identities of users and admins
Identity(enrollment_id, user)	Class represents a tuple containing

class hfc.fabric_network.wallet.**FileSystenWallet**(*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.**DESCRIPTOR**

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.**DESCRIPTOR**

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.**DESCRIPTOR**

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.**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.**DESCRIPTOR**

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.**DESCRIPTOR**

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.**DESCRIPTOR**

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_MSPSENTRY**

hfc.protos.discovery.protocol_pb2.**_CONFIGRESULT_ORDERERSENTRY**

hfc.protos.discovery.protocol_pb2.**_CONFIGRESULT**

hfc.protos.discovery.protocol_pb2.**_PEERMEMBERSHIPQUERY**

hfc.protos.discovery.protocol_pb2.**_PEERMEMBERSHIPRESULT_PEERSBYORGENTRY**

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_ENDORSERSBYGROUPSENTRY**

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

DiscoveryStub(channel)	Discovery defines a service that serves information about the fabric network
DiscoveryServicer()	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, chaincodes, 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, chaincodes, 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.**DESCRIPTOR**

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.**_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.**message_type**

hfc.protos.gossip.message_pb2.**containing_oneof**

hfc.protos.gossip.message_pb2.**enum_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.**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.**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.**containing_type**
hfc.protos.gossip.message_pb2.**containing_oneof**
hfc.protos.gossip.message_pb2.**containing_oneof**
hfc.protos.gossip.message_pb2.**containing_oneof**
hfc.protos.gossip.message_pb2.**containing_oneof**
hfc.protos.gossip.message_pb2.**containing_oneof**
hfc.protos.gossip.message_pb2.**containing_oneof**
hfc.protos.gossip.message_pb2.**containing_oneof**
hfc.protos.gossip.message_pb2.**containing_oneof**
hfc.protos.gossip.message_pb2.**containing_oneof**
hfc.protos.gossip.message_pb2.**containing_oneof**
hfc.protos.gossip.message_pb2.**containing_oneof**
hfc.protos.gossip.message_pb2.**containing_oneof**
hfc.protos.gossip.message_pb2.**containing_oneof**
hfc.protos.gossip.message_pb2.**containing_oneof**
hfc.protos.gossip.message_pb2.**containing_oneof**
hfc.protos.gossip.message_pb2.**containing_oneof**
hfc.protos.gossip.message_pb2.**containing_oneof**
hfc.protos.gossip.message_pb2.**containing_oneof**
hfc.protos.gossip.message_pb2.**containing_oneof**
hfc.protos.gossip.message_pb2.**containing_oneof**
hfc.protos.gossip.message_pb2.**containing_oneof**
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.**enum_type**
hfc.protos.gossip.message_pb2.**enum_type**
hfc.protos.gossip.message_pb2.**message_type**
hfc.protos.gossip.message_pb2.**enum_type**
hfc.protos.gossip.message_pb2.**enum_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.**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

GossipStub(channel)	Gossip
GossipServicer()	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

Submodules

hfc.protos.idemix.idemix_pb2

Module Contents

hfc.protos.idemix.idemix_pb2.**_b**

hfc.protos.idemix.idemix_pb2.**_sym_db**

hfc.protos.idemix.idemix_pb2.**DESCRIPTOR**

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.**message_type**

hfc.protos.idemix.idemix_pb2.**message_type**

hfc.protos.idemix.idemix_pb2.**message_type**

hfc.protos.idemix.idemix_pb2.**message_type**

hfc.protos.idemix.idemix_pb2.**message_type**

hfc.protos.idemix.idemix_pb2.**message_type**

hfc.protos.idemix.idemix_pb2.**message_type**

hfc.protos.idemix.idemix_pb2.**message_type**

hfc.protos.idemix.idemix_pb2.**message_type**

hfc.protos.idemix.idemix_pb2.**message_type**

hfc.protos.idemix.idemix_pb2.**message_type**

hfc.protos.idemix.idemix_pb2.**message_type**

hfc.protos.idemix.idemix_pb2.**message_type**

hfc.protos.idemix.idemix_pb2.**message_type**

hfc.protos.idemix.idemix_pb2.**message_type**

hfc.protos.idemix.idemix_pb2.**message_type**

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.**DESCRIPTOR**

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.**DESCRIPTOR**
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.**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.**DESCRIPTOR**

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.**DESCRIPTOR**

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.**DESCRIPTOR**

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.**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.**FabricOUIdentifier**

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.**DESCRIPTOR**

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.**DESCRIPTOR**

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 mashaled SeekInfo 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

ClusterStub(channel)	Cluster defines communication between cluster members.
ClusterServicer()	Cluster defines communication between cluster members.

Functions

add_ClusterServicer_to_server(servicer, server)

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.**DESCRIPTOR**

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.**DESCRIPTOR**

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.**DESCRIPTOR**
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.**DESCRIPTOR**
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

AdminStub(channel)	Interface exported by the server.
AdminServicer()	Interface exported by the server.

Functions

add_AdminServicer_to_server(servicer, server)

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.**DESCRIPTOR**

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.**DESCRIPTOR**

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.**_CDSDATA**

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.**CDSData**

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.**DESCRIPTOR**

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

ChaincodeSupportStub(channel)	Interface that provides support to chaincode execution. ChaincodeContext
ChaincodeSupportServicer()	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.**DESCRIPTOR**

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.**DESCRIPTOR**

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.**DESCRIPTOR**

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.**DESCRIPTOR**

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.**DESCRIPTOR**

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.**DESCRIPTOR**

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.**DESCRIPTOR**

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.**DESCRIPTOR**

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.**DESCRIPTOR**

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.**DESCRIPTOR**

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.**DESCRIPTOR**

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.**DESCRIPTOR**

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.**message_type**

hfc.protos.token.prover_pb2.**message_type**

hfc.protos.token.prover_pb2.**message_type**

hfc.protos.token.prover_pb2.**message_type**

hfc.protos.token.prover_pb2.**message_type**

hfc.protos.token.prover_pb2.**message_type**

hfc.protos.token.prover_pb2.**message_type**

hfc.protos.token.prover_pb2.**message_type**

hfc.protos.token.prover_pb2.**message_type**

hfc.protos.token.prover_pb2.**message_type**

hfc.protos.token.prover_pb2.**message_type**

hfc.protos.token.prover_pb2.**message_type**

hfc.protos.token.prover_pb2.**message_type**

hfc.protos.token.prover_pb2.**message_type**

hfc.protos.token.prover_pb2.**message_type**

hfc.protos.token.prover_pb2.**message_type**

hfc.protos.token.prover_pb2.**message_type**

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.**containing_oneof**

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.**message_type**

hfc.protos.token.prover_pb2.**message_type**

hfc.protos.token.prover_pb2.**message_type**

hfc.protos.token.prover_pb2.**message_type**

hfc.protos.token.prover_pb2.**message_type**

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

ProverStub(channel)	Prover provides support to clients for the creation of FabToken transactions,
ProverServicer()	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 succeffully 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.**DESCRIPTOR**

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.**DESCRIPTOR**
hfc.protos.transientstore.transientstore_pb2.**_TXPVTREADWRITESETWITHCONFIGINFO_COLLECTIONCO**

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

Key()	An abstract base class for Key.
AsymmetricKey()	An asymmetric key.
Crypto()	An abstract base class for crypto.
Ecies(security_level=CURVE_P_256_Size, hash_algorithm=SHA2)	A crypto implementation based on ECDSA and SHA.

Functions

generate_nonce(size)	Generate a secure random for cryptographic use.
ecies(security_level=CURVE_P_256_Size, hash_algorithm=SHA2)	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

NullHandler(level=NOTSET)	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

create_grpc_channel(target, cert_file=None, client_key=None, client_cert=None, opts=None)	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

KeyValueStore()	An key value store for blockchain application state persistence.
FileKeyValueStore(path)	A key value store implementation based file system.

Functions

_make_dir(path)	
file_key_value_store(path)	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 common.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

proto_str(x)	
create_serialized_identity(user)	Create serialized identity from user.
build_header(creator, channel_header, nonce)	This function will build the common header.
build_channel_header(type, tx_id, channel_id, timestamp, epoch=0, extension=None, tls_cert_hash=None)	Build channel header.
string_to_signature(string_signatures)	Check if signatures are already in protobuf format.
current_timestamp()	Get current timestamp.
extract_channel_config(configtx_proto_envelope)	Extracts the protobuf 'ConfigUpdate' object out ouf the 'ConfigEnvelope'.
build_cc_proposal(cci_spec, header, transient_map)	Create an chaincode transaction proposal
sign_proposal(tx_context, proposal)	Sign a proposal
send_transaction_proposal(proposal, tx_context, peers)	Send transaction proposal
send_transaction(orderers, tran_req, tx_context)	Send a transaction to the chain's orderer service (one or more
sign_tran_payload(tx_context, tran_payload_bytes)	Sign a transaction payload
build_tx_req(ProposalResponses)	Check the endorsements from peers
send_install_proposal(tx_context, peers)	Send install chaincode proposal
zeroTarInfo(tarinfo)	
package_chaincode(cc_path, cc_type=CC_TYPE_GOLANG)	Package all chaincode env into a tar.gz file
pem_to_der(pem)	
stream_envelope(envelope)	

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 ouf the 'ConfigEnvelope'.

Parameters **configtx_proto_envelope** (*common_pb2.Envelope*) – The encoded bytes of the ConfigEnvelope protofbuf.

Returns (config_update) The encoded bytes of the ConfigUpdate protobuf, ready to be signed

Return type configtx_pb2.ConfigUpadeEnvelope.config_update

Raises **ValueError** – If there is an error in protobuf_decode due to a wrong or not valid profobuf 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

INDEX

Symbols

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

Z