chaincode

Bitcoin Core Functional Test Framework

Content

Part 1: Where are we?

Part 2: Test Framework

Part 3: What's in a test?

Part 4: Example test

Part 5: Missing parts

Part 1: Where are we?

What are functional tests?

Test the application from user/network perspective

In Bitcoin Core: Interactions of user and other nodes through RPC and P2P interfaces

Testing the full stack

Generally slower than unit tests

When do you add/edit functional tests?

Full Features/Functionality that uses multiple layers of the stack!

RPC inputs/outputs

Logic/behavior relevant for the user

P2P/network behavior

Where are the files?

`test/*` NOT `src/test/*`

Areas

- `feature_*` (full features that are not wallet/mining/mempool)
- `interface_*` (REST, ZMQ etc.)
- `mempool_*`
- `mining_*`
- `p2p_*`
- `rpc_*`
- `tool_*` (tool_wallet.py)
- `wallet_*`

Running tests

Directly (shows INFO log outputs)

- `test/functional/feature_rbf.py`

Indirectly through test harness (no INFO log outputs)

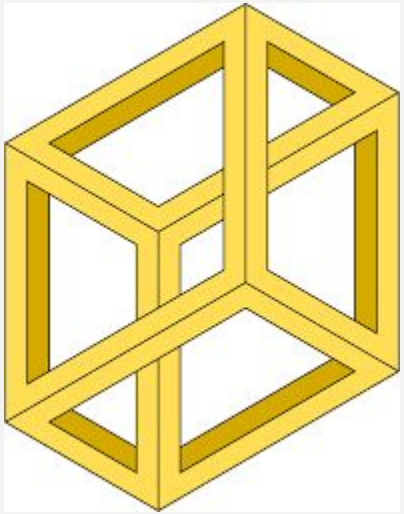
- `test/functional/test_runner.py feature_rbf.py/`
- `test/functional/test_runner.py test/functional/wallet*`

All (parallel)

- `test/functional/test_runner.py`

Options (e.g. `--trace-rpc`, `--nocleanup` etc.)

Part 2: Test Framework



`test/test_framework/*` (selection)

`util.py`: asserts and other helper functions

`test_framework.py`: mainly `BitcoinTestFramework` class

`key.py`: ECC math classes and functions (OpenSSL EC_Key wrapper)

`script.py`: utilities for working with transaction scripts

`blocktools.py`: helpers for creating blocks and transactions

`mininode.py`: P2P connectivity helpers, `P2PDataStore()` etc.

Part 3: What's in a test?

Documentation and Logs

To explain what you are doing:

Docstrings

Comments

```
self.log.info(...)
```

Test class

Test is a subclass of `BitcoinTestFramework`

Overrides

- `set_test_params()` override test parameters
- `run_test()` override for actual test case
- others for custom setup (see `test_framework/test_framework.py`)

Node calls

```
self.nodes[0].add_p2p_connection(BaseNode())
```

Most RPC calls are undefined methods

Regtest RPC

- `generate()` etc.

Wait methods

- `waitforblockheight()` etc.

Global methods like connect nodes

P2P introspection

`sync_all()`, `sync_blocks()` etc.

Subclass of `P2PInterface` with `on_*` hook method overrides

Used to keep a P2P connection to the node under test

Evaluate messages the node sends out

Examples:

- `on_block()`
- `on_inv()`

Part 4: Examples

Getblockchaininfo: https://github.com/bitcoin/bitcoin/blob/master/test/functional/rpc_blockchain.py

Part 5: MISC

Debugging and Logging

Use python debugger (pdb)

```
import pdb; pdb.set_trace()
```

Attach `pdb`/`lldb` to bitcoind instance

Consolidate logs with `combine_logs.py`

Logs in temp folder

```
/path/to/bitcoin/test/functional/combine_logs.py
'/var/folders/9z/n7rz_6cj3bq__11k5kcrsvvm0000gn/T/bitcoin_func_test_7n
eje5nv'
```

Get started!

Further reading

- `test/README.md`
- `test/functional/README.md`
- `test/functional/example_test.py`

TODOs for you

- 39 open issues with label “tests”: https://github.com/bitcoin/bitcoin/issues?utf8=%E2%9C%93&q=is%3Aissue+is%3Aopen+label%3Atests
- Improve test coverage: https://marcofalke.github.io/btc_cov/

Thank you and questions?