pytest-ethereum Documentation

Release 0.1.3-alpha.7

Jason Carver

Jun 24, 2019

Contents

Pytest library for ethereum projects.

CHAPTER 1

Contents

Warning: Pytest-Ethereum is still under active development, and yet to reach a stable release. It should not be used in production yet.

1.1 Overview

This library is designed to make deploying and testing smart contracts simple, using *Py-EthPM* and *pytest*.

1.1.1 Deployer

This library exposes a `Deployer` fixture to help create contract instances for any contract types available in the manifest that generated the `Deployer` instance. To create a `Deployer` instance, you must provide a `pathlib.Path` object pointing towards a valid manifest according to the EthPM Specification.

To deploy any of the available *contract types* onto the default `w3` instance, simply call `deploy` on the deployer and a newly created `Package` instance (which contains the newly created contract instance in its *deployments*) will be returned, along with the address of the newly deployed contract type.

```
from pathlib import Path
from ethpm import Package
from eth_utils import is_address

@pytest.fixture
def owned_deployer(deployer):
    owned_manifest_path = Path('to/owned/manifest.json')
    return deployer(owned_manifest_path)

def test_owned_contract(owned_deployer)
    owned_package = owned_deployer.deploy("owned")
    assert isinstance(owned_package, Package)
```

(continues on next page)

(continued from previous page)

```
    owned_contract_instance = owned_package.deployments.get_deployment_instance("Owned
↪")
    assert is_address(owned_contract_instance.address)
```

`Deployer.`**`deploy`**(*contract_type*)

Returns a `Package` instance, containing a freshly deployed instance of the given *contract_type* (if sufficient data is present in the manifest). To add transaction kwargs (i.e. "from"), pass them in as a dict to the `transaction` keyword.

```
deploy("Contract", arg1, transaction={"from": web3.eth.accounts[1]})
```

`Deployer.`**`register_strategy`**(*contract_type*, *strategy*)

If a *contract_type* requires linking, then you *must* register a valid strategy constructed with the `Linker` before you can deploy an instance of the *contract_type*.

1.1.2 Linker

If a contract factory requires linking, you must register a "strategy" for a particular contract factory with the deployer. It is up to you to design an appropriate strategy for a contract factory.

Three `linker` functions are made available:

`deploy`(*contract_name*, **args=None*)

To deploy an instance of *contract_name*. If the contract constructor requires arguments, they must also be passed in.

`link`(*contract_name*, *linked_type*)

Links a *contract_name* to a *linked_type*. The *linked_type* must have already been deployed.

`run_python`(*callback_fn*)

Calls any user-defined *callback_fn* on the contracts available in the active *Package*. This can be used to call specific functions on a contract if they are part of the setup. Returns the original, unmodified *Package* that was passed in.

For example, the *Escrow* contract factory requires linking to an instance of the *SafeSendLib* before an *Escrow* contract instance can be deployed. This is how you would set up a strategy for *Escrow*

```
@pytest.fixture
def escrow_deployer(deployer, manifest_dir):
    escrow_manifest_path = manifest_dir / "escrow_manifest.json"
    return deployer(escrow_manifest_path)


@pytest.fixture
def escrow_contract_instance(escrow_deployer, w3):
    escrow_strategy = linker(
        deploy("SafeSendLib"),
        link("Escrow", "SafeSendLib"),
        deploy("Escrow", w3.eth.accounts[0]),
    )
    escrow_deployer.register_strategy("Escrow", escrow_strategy)
    linked_escrow_package, _ = escrow_deployer.deploy("Escrow")
    return linked_escrow_package.deployments.get_deployment("Escrow")
```

1.1.3 Log

The `Log` class is available to help with testing for contract events, and the contents of the emitted logs.

`tests/fixtures/ping.vy`

```
Ping: event({first: indexed(bytes32), second: bytes32})

@public
def __init__():
    pass

@public
def ping(_first: bytes32, _second: bytes32):
    log.Ping(_first, _second)
```

```
# SETUP
ping_package = deployer.deploy("ping")
ping_instance = ping_package.deployments.get_contract_instance("ping")
tx_hash = ping_instance.functions.ping(b"one", b"two")
receipt = w3.eth.waitForTransactionReceipt(tx_hash)
```

1.2 Release Notes

1.2.1 v0.1.0-alpha.1

- Launched repository, claimed names for pip, RTD, github, etc

CHAPTER 2

Indices and tables

- genindex
- modindex

Index