

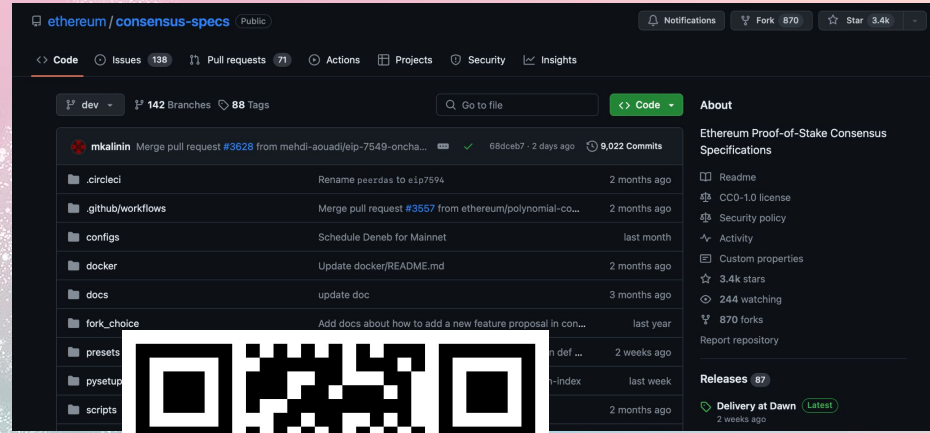
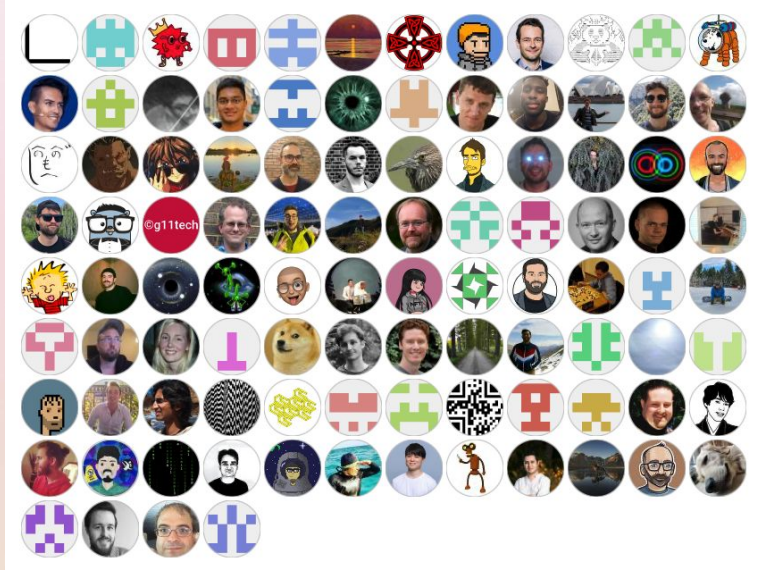
How to use Executable Consensus Pyspec

Hsiao-Wei Wang

Ethereum Foundation - Consensus R&D

2024-03-25 version

<https://github.com/ethereum/consensus-specs>



CC0-1.0 license

1. It's a collection of Ethereum core **consensus specifications**

Define the consensus protocol that running by the consensus layer (CL) clients



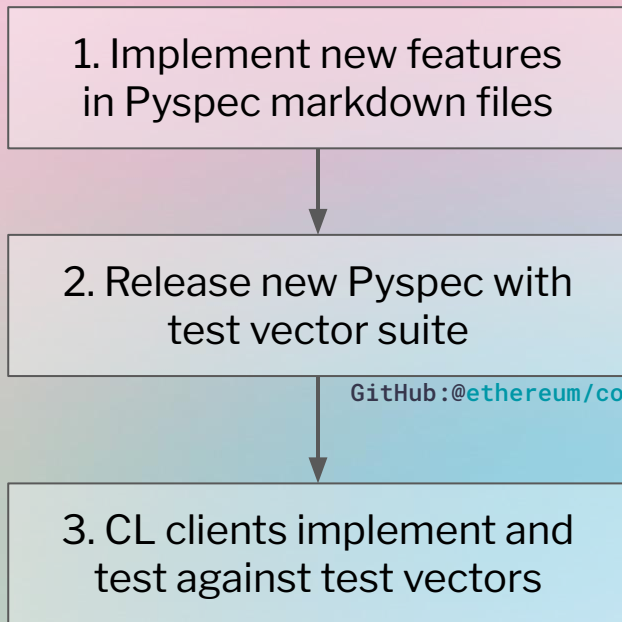
2. It's **executable** and **verifiable**

It can be built into a Python program that can be executed

3. It's **test vector generator**

It can generate the test vectors for CL clients to run with and test against the consensus rules.

Adding New Feature Patch



GitHub: [@ethereum/consensus-spec-tests](https://github.com/ethereum/consensus-spec-tests)

Python is very readable to developers



Credits: r/ProgrammerHumor u/Cant_Grow_a_Stasch [“Python == pseudocode”](#)

How to read it?

/specs/

- **_features**
 - **eip6110**
 - ...
- **altair**
- **bellatrix**
- **capella**
- **deneb**
- **phase0**

Mainnet protocol upgrades

WIP features

How to read it?

/specs/

- altair
- bellatrix
- phase0

Markdown files

- **beacon-chain.md**
- deposit-contract.md
- fork-choice.md
- p2p-interface.md
- validator.md
- weak-subjectivity.md

Type and Values Definitions

Notation

Code snippets appearing in `this style` are to be interpreted as Python 3 code.

Custom types

We define the following Python custom types for type hinting and readability:

Name	SSZ equivalent	Description
Slot	uint64	a slot number
Epoch	uint64	an epoch number
CommitteeIndex	uint64	a committee index at a slot
ValidatorIndex	uint64	a validator registry index

Constants

The following values are (non-configurable) constants used throughout the specification.

Misc

Name	Value
GENESIS_SLOT	Slot(0)
GENESIS_EPOCH	Epoch(0)
FAR_FUTURE_EPOCH	Epoch($2^{64} - 1$)

Preset

Note: The below configuration is bundled as a preset: a bundle of configuration variables which are expected to differ between different modes of operation, e.g. testing, but not generally between different networks. Additional preset configurations can be found in the [configs](#) directory.

Misc

Name	Value
MAX_COMMITTEES_PER_SLOT	uint64(2^{*6}) (= 64)
TARGET_COMMITTEE_SIZE	uint64(2^{*7}) (= 128)
MAX_VALIDATORS_PER_COMMITTEE	uint64(2^{*11}) (= 2,048)
SHUFFLE_ROUND_COUNT	uint64(90)

Configuration

Note: The default mainnet configuration values are included here for illustrative purposes. Defaults for this more dynamic type of configuration are available with the presets in the [configs](#) directory. Testnets and other types of chain instances may use a different configuration.

Genesis settings

Name	Value
MIN_GENESIS_ACTIVE_VALIDATOR_COUNT	uint64(2^{*14}) (= 16,384)
MIN_GENESIS_TIME	uint64(1606824000) (Dec 1, 2020, 12pm UTC)

SSZ Containers

Note: we also use SSZ **hash tree root** as the digests of consensus objects.

See:

<https://github.com/ethereum/consensus-specs/blob/dev/ssz/simple-serialize.md>

BeaconBlockBody

```
class BeaconBlockBody(Container):
    randao_reveal: BLSSignature
    eth1_data: Eth1Data # Eth1 data vote
    graffiti: Bytes32 # Arbitrary data
    # Operations
    proposer_slashings: List[ProposerSlashing, MAX_PROPOSER_SLASHINGS]
    attester_slashings: List[AttesterSlashing, MAX_ATTESTER_SLASHINGS]
    attestations: List[Attestation, MAX_ATTESTATIONS]
    deposits: List[Deposit, MAX_DEPOSITS]
    voluntary_exits: List[SignedVoluntaryExit, MAX_VOLUNTARY_EXITS]
```

BeaconBlock

```
class BeaconBlock(Container):
    slot: Slot
    proposer_index: ValidatorIndex
    parent_root: Root
    state_root: Root
    body: BeaconBlockBody
```

State transition function

All are “pure” functions.

Beacon chain state transition function

The post-state corresponding to a pre-state `state` and a signed block `signed_block` is defined as `state_transition(state, signed_block)`. State transitions that trigger an unhandled exception (e.g. a failed `assert` or an out-of-range list access) are considered invalid. State transitions that cause a `uint64` overflow or underflow are also considered invalid.

```
def state_transition(state: BeaconState, signed_block: SignedBeaconBlock, validate_result: bool=True) -> None:
    block = signed_block.message
    # Process slots (including those with no blocks) since block
    process_slots(state, block.slot)
    # Verify signature
    if validate_result:
        assert verify_block_signature(state, signed_block)
    # Process block
    process_block(state, block)
    # Verify state root
    if validate_result:
        assert block.state_root == hash_tree_root(state)
```



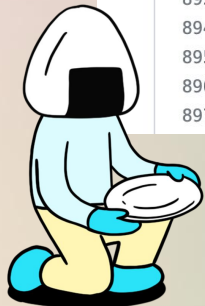
```
post_state = state_transition(pre_state, block)
```

Useful resources to understand CL

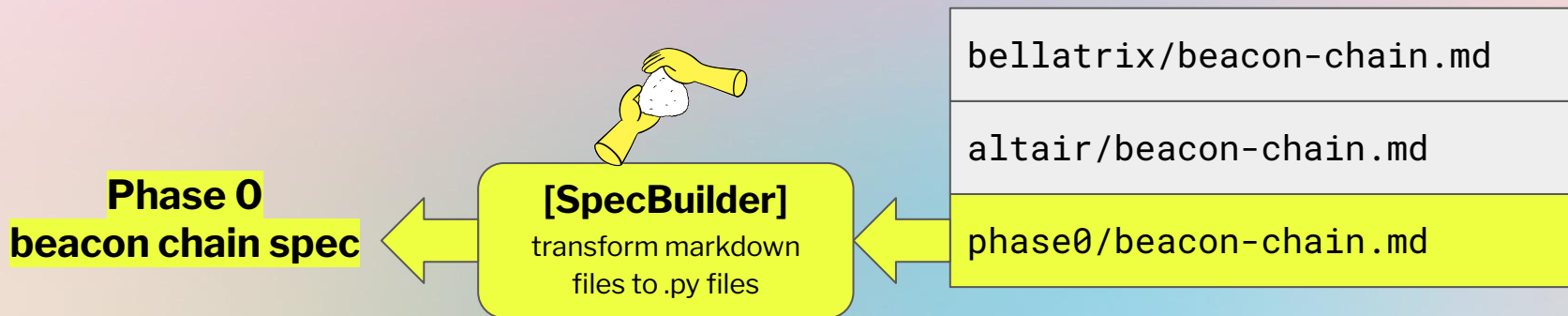
- Vitalik Buterin's Annotated spec: <https://github.com/ethereum/annotated-spec>
- Ben Edgington's Upgrading Ethereum Book: <https://eth2book.info>

The elf in setup.py

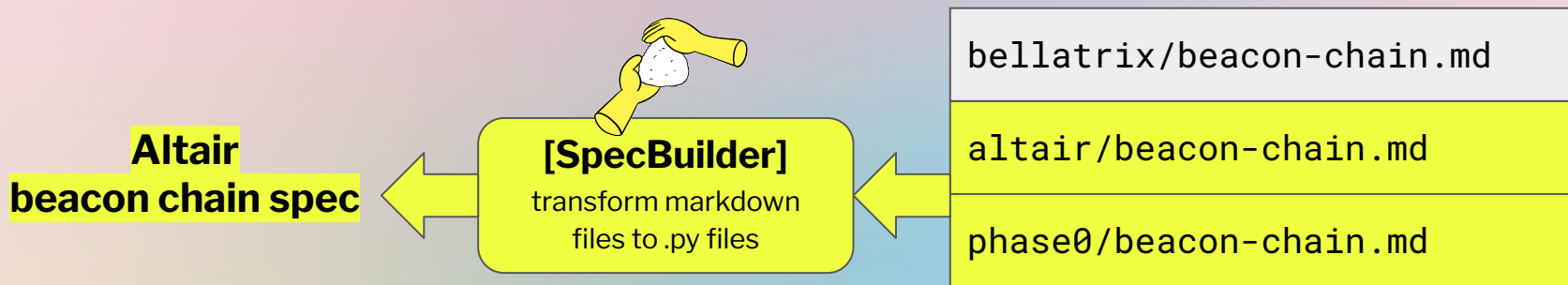
```
879 def _build_spec(preset_name: str, fork: str,  
880                 source_files: Sequence[Path], preset_files: Sequence[Path], config_file: Path) -> str:  
881     preset = load_preset(preset_files)  
882     config = load_config(config_file)  
883     all_specs = [get_spec(spec, preset, config) for spec in source_files]  
884  
885     spec_object = all_specs[0]  
886     for value in all_specs[1:]:  
887         spec_object = combine_spec_objects(spec_object, value)  
888  
889     class_objects = {**spec_object.ssz_objects, **spec_object.dataclasses}  
890  
891     # Ensure it's ordered after multiple forks  
892     new_objects = {}  
893     while OrderedDict(new_objects) != OrderedDict(class_objects):  
894         new_objects = copy.deepcopy(class_objects)  
895         dependency_order_class_objects(class_objects, spec_object.custom_types)  
896  
897     return objects_to_spec(preset_name, spec_object, spec_builders[fork], class_objects)
```



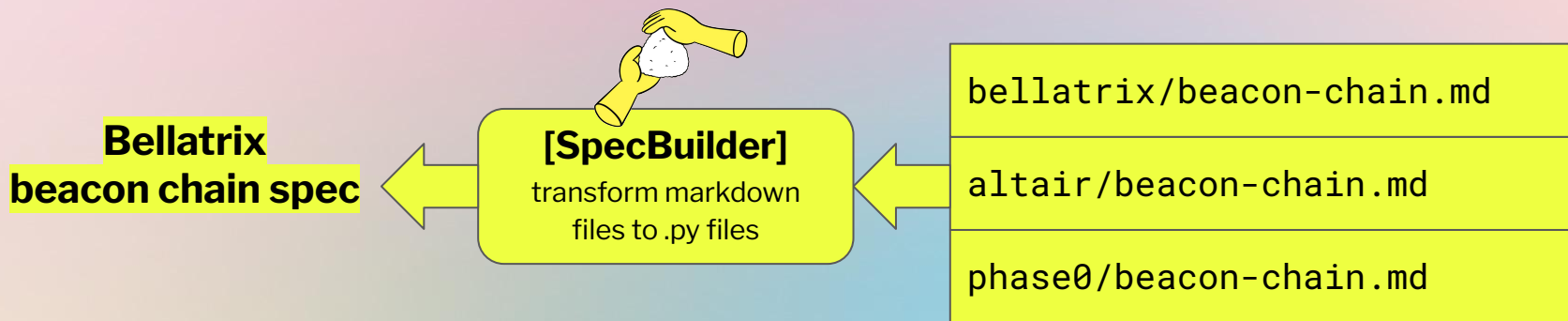
Extend the previous hard forks



Extend the previous hard forks



Extend the previous hard forks





How to use Pyspec?

Installation (Python3.8+)

Install from source with venv:

- a. Download the source code:

```
git clone https://github.com/ethereum/consensus-specs.git
```

- b. Install with Makefile commands

```
cd consensus-specs
```

```
make install_test && make pyspec
```

Run your first pyspec program!

```
>> from eth2spec.bellatrix import mainnet as spec
```

```
>> hello = b"Hello World"
```

```
>> body = spec.BeaconBlockBody(
```

```
>>     graffiti=hello + b'\0' * (32 - len(hello))
```

```
>> )
```

```
>> block = spec.BeaconBlock(body=body)
```

```
>> print(block.body.graffiti.decode("utf-8"))
```

```
Hello World
```

Write your first pyspec test case!

```
@with_all_phases
@spec_state_test
def test_empty_block_transition(spec, state):
    pre_slot = state.slot
    pre_eth1_votes = len(state.eth1_data_votes)
    pre_mix = spec.get_randao_mix(state, spec.get_current_epoch(state))

    yield 'pre', state

    block = build_empty_block_for_next_slot(spec, state)
    signed_block = state_transition_and_sign_block(spec, state, block)

    yield 'blocks', [signed_block]
    yield 'post', state

    assert len(state.eth1_data_votes) == pre_eth1_votes + 1
    assert spec.get_block_root_at_slot(state, pre_slot) == signed_block.message.parent_root
    assert spec.get_randao_mix(state, spec.get_current_epoch(state)) != pre_mix
```

Write your first pyspec test case!

```
@with_all_phases
@spec_state_test
def test_empty_block_transition(spec, state):
    pre_slot = state.slot
    pre_eth1_votes = len(state.eth1_data_votes)
    pre_mix = spec.get_randao_mix(state, spec.get_current_epoch(state))

    yield 'pre', state

    block = build_empty_block_for_next_slot(spec, state)
    signed_block = state_transition_and_sign_block(spec, state, block)

    yield 'blocks', [signed_block]
    yield 'post', state

    assert len(state.eth1_data_votes) == pre_eth1_votes + 1
    assert spec.get_block_root_at_slot(state, pre_slot) == signed_block.message.parent_root
    assert spec.get_randao_mix(state, spec.get_current_epoch(state)) != pre_mix
```

Prepare the block

Write your first pyspec test case!

```
@with_all_phases
@spec_state_test
def test_empty_block_transition(spec, state):
    pre_slot = state.slot
    pre_eth1_votes = len(state.eth1_data_votes)
    pre_mix = spec.get_randao_mix(state, spec.get_current_epoch(state))

    yield 'pre', state

    block = build_empty_block_for_next_slot(spec, state)
    signed_block = state_transition_and_sign_block(spec, state, block)

    yield 'blocks', [signed_block]
    yield 'post', state

    assert len(state.eth1_data_votes) == pre_eth1_votes + 1
    assert spec.get_block_root_at_slot(state, pre_slot) == signed_block.message.parent_root
    assert spec.get_randao_mix(state, spec.get_current_epoch(state)) != pre_mix
```

Verify the post-state
with assertions



Pyspec as the **test vector** generator

```
@with_all_phases
@spec_state_test
def test_empty_block_transition(spec, state):
    pre_slot = state.slot
    pre_eth1_votes = len(state.eth1_data_votes)
    pre_mix = spec.get_randao_mix(state, spec.get_current_epoch(state))

    yield 'pre', state

    block = build_empty_block_for_next_slot(spec, state)
    signed_block = state_transition_and_sign_block(spec, state, block)

    yield 'blocks', [signed_block]
    yield 'post', state

    assert len(state.eth1_data_votes) == pre_eth1_votes + 1
    assert spec.get_block_root_at_slot(state, pre_slot) == signed_block.message.parent_root
    assert spec.get_randao_mix(state, spec.get_current_epoch(state)) != pre_mix
```

Yield test vectors



Pyspec as the **test vector** generator

```
@with_all_phases
@spec_state_test
def test_empty_block_transition(spec, state):
    pre_slot = state.slot
    pre_eth1_votes = len(state.eth1_data_votes)
    pre_mix = spec.get_randao_mix(state, spec.get_current_epoch)

    yield 'pre', state

    block = build_empty_block_for_next_slot(spec, state)
    signed_block = state_transition_and_sign_block(spec, state, block)

    yield 'blocks', [signed_block]
    yield 'post', state

    assert len(state.eth1_data_votes) == pre_eth1_votes + 1
    assert spec.get_block_root_at_slot(state, pre_slot) == sig
    assert spec.get_block_root_at_slot(state, pre_slot) == sig
```

ethereum / consensus-spec-tests Public

<> Code Issues 9 Pull requests Discussions ...

master consensus-spec-tests / tests / minimal / phase0 / sanity / blocks / pyspec_tests / empty_block_transition /

hwwhww release v1.2.0-rc.1 tests on May 24 History

..	
blocks_0.ssz_snappy	5 months ago
meta.yaml	3 years ago
post.ssz_snappy	5 months ago
pre.ssz_snappy	5 months ago

```
post_state = state_transition(pre_state, block)
```

Documents

- Pyspec:

<https://github.com/ethereum/consensus-specs/blob/dev/tests/README.md>

- Test formats:

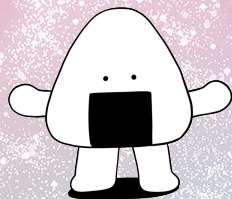
<https://github.com/ethereum/consensus-specs/blob/dev/tests/formats/README.md>

How to contribute to pyspec?

- Level 1: 👁️ Look through the specifications files to learn about the specifications logic and help review it
- Level 2: 🦆 Help refactor the codebase
- Level 3: 😈 Try to hack some new edge test cases!
- Level 4: 💰 Submit to bug bounty (<https://ethereum.org/en/bug-bounty/>)



Thank you!



Hsiao-Wei Wang

Researcher, Ethereum Foundation

`hww@ethereum.org`

Onigiri images: furiirakun.com