Background

The Bitcoin BIP process was originally based on Python's PEP process. Other communities, including Ethereum and Zcash in turn based theirs upon the BIP process, so indirectly they, too, inherit from PEP.

Python (https://www.python.org/dev/peps/pep-0001/)

- The "great-granddaddy" of the proposal processes. Modelled after the RFC process.
- PEP-1 created 13-Jun-2000
- PEPs are "the primary mechanisms for proposing major new features, for collecting community input on an issue, and for documenting the design decisions that have gone into Python"
- Three types:
    - Standards Track
    - Informational
    - Process
- Four core roles:
    - Steering council: "the Steering Council's design authority derives from their election by the currently active core developers"
    - Core developers
    - "BDFL" ("PEP decision makers") whose authority ultimately derives from the original "Benevolent Dictator for Life" (Guido).
    - PEP editors: "PEP editorship is by invitation of the current editors"
- All PEP workflow is conducted via GitHub.
- Criteria for acceptance
    - "For a PEP to be accepted it must meet certain minimum criteria. It must be a clear and complete description of the proposed enhancement. The enhancement must represent a net improvement. The proposed implementation, if applicable, must be solid and must not complicate the interpreter unduly. Finally, a proposed enhancement must be "pythonic" in order to be accepted by the Steering Council. (However, "pythonic" is an imprecise term; it may be defined as whatever is acceptable to the Steering Council. This logic is intentionally circular.)"
    - "The PEP editors will not unreasonably deny publication of a PEP. Reasons for denying PEP status include duplication of effort, being technically unsound, not providing proper motivation or addressing backwards compatibility, or not in keeping with the Python philosophy."
- Status: there is an explicit Rejected status.
- "The final authority for PEP approval is the Steering Council. However, whenever a new PEP is put forward, any core developer that believes they are suitably experienced to make the final decision on that PEP may offer to serve as the BDFL-Delegate for that PEP, and they will then have the authority to approve (or reject) that PEP. Individuals taking on this responsibility are free to seek additional

guidance from the Steering Council at any time, and are also expected to take the advice and perspectives of other core developers into account."

- Standards track PEPs must be discussed on the python-dev mailing list.
- Much more comprehensive governance is described in Python Language Governance. Of particular interest is the fact that there is an explicit, maintained list of developers who may vote to elect the council. Thus ultimate legitimacy and authority derive from these developers, collectively.

Bitcoin (https://github.com/bitcoin/bips/blob/master/bip-0002.mediawiki)

- Originally based on BIP1, Luke Dashjr made improvements ("a more well-defined and clear process") in BIP2.
- The revision history ("historical record") of a BIP is its git history.
- Three types: Standard, Informational, Process (same as PEP).
- "It is highly recommended that a single BIP contain a single key proposal or new idea. The more focused the BIP, the more successful it tends to be. If in doubt, split your BIP into several well-focused ones."
- Reflects the informal nature of Bitcoin governance (as opposed to Python). BIP2 makes no pretense of being a governance document. It explicitly states that it only governs the "Status" field of each BIP, and *tries to be descriptive rather than normative or prescriptive in nature.*
- The basic workflow is that anyone is free to champion a proposal, that proposals should be discussed and vetted on the bitcoin-devs mailing list before being turned into BIPs, and that there are very strict gates and guidelines based on the real, observed behavior in the Bitcoin community, network, and "economy." E.g., a BIP needs a 95% supermajority (via miner votes) to proceed to the Final status. This reenforces the idea that BIPs are intended to *reflect reality* rather than be prescriptive.
- Has an explicit Rejected status—but reasons for rejection are vague, including "not in keeping with the Bitcoin philosophy."
- References "Community" 15 times, including as part of the criteria for moving a BIP to Proposed: this requires that the "community plans to progress it to the Final status."
- Contains a list of explicitly approved licenses for BIPs; code may be licensed separately from BIP content.

Zcash (https://github.com/zcash/zips, https://zips.z.cash/zip-0000)

- A fork of BIP2. Not too many changes.
- Very little explicit meta-governance ("Process") is specified via ZIP. The one exception was the debate last year about the founder's reward and dev fund, for which a slew of Process ZIPs were submitted.
- Changes include:
  - Adds a Consensus category (as distinct from Standards which don't affect consensus)
  - Adds an explicit Implemented status
  - Adds a required section "Security and privacy considerations"

    - A long, explicit list of reasons for rejection (but still includes “not in keeping with the Zcash philosophy”)
    - Different numbering conventions
    - Requires a minimum of two ZIP editors: one from ECC and one from the Foundation

Ethereum (https://eips.ethereum.org/EIPS/eip-1, https://eips.ethereum.org/)

- Ethereum has only a single proposals process. All proposals, whether related to technology, governance, or anything else, flow through the EIP process.
- The EIP process, and the related All Core Devs process, are the only formal governance mechanism of Ethereum.
- Anyone is free to submit a new EIP at any time, as a GitHub issue at https://github.com/ethereum/EIPs/. There are loose requirements for what an EIP should include: such as rationale, information on backwards compatibility, test cases, security considerations, etc.
- There are three main types of EIP: Standard, Meta, and Informational (similar to PEP/BIP). Standard Track includes Core, Networking, Interface, and ERC. Note that ERC proposes a standard but does not directly propose a code/protocol change, so it’s a bit misleading that it be grouped together with the other Standard track proposals.
- EIPs have a rather complex workflow that is not well understood even by core contributors and EIP editors. There is often meta-debate about whether a particular EIP status change is allowed, or required, in the first place.
- The gist of the workflow is that there is a series of “gates”, beginning with EIP authorship, up through full adoption and implementation by all clients, and finally inclusion in a future network upgrade. At each of these “gates” a distinct group—EIP authors, EIP editors, All Core Devs, individual dev teams, and, ultimately, node operators—may choose to effectively veto a proposal by not acting upon it, but no group has individual authority to “ratify” a proposal.
- The process by which an EIP is brought up on an All Core Devs call, decided upon, and accepted or rejected is arcane, informal, and not documented.
- Any correctly formatted EIP should be merged by an editor and marked Accepted. In some cases the editors are reluctant to do so if an EIP is especially contentious. In other words, there is a "pocket veto" option.
- Just because an EIP is “Accepted” doesn’t mean it will be implemented. Just because it’s implemented in one client doesn’t mean that others will implement it. In practice, however, all client teams do universally implement the same things.
- It’s unclear how someone becomes an EIP editor or a “core dev.” This process is undocumented and murky. There is no explicit list of core devs. In practice, it’s whomever Hudson invites to join the calls.
- Core devs are free to ignore an EIP if they wish, and to implement things that are not actually EIPs, although in practice this is extremely rare. (In this respect, EIPs resemble BIPs and the aim of the BIP process to be descriptive rather than normative.)

- No explicit Rejected status. Can only deny/defer promotion (to “Draft”). Like Bitcoin, this is ill-defined: reasons include “not in keeping with the Ethereum philosophy.” As a result, in practice, zombie EIPs tend to stick around from time to time
- There's a lot of contention around how "the community" feels about this or that EIP, and around the meta question of how we know what the community thinks, whether and to what extent we care about this, etc.
- There is no official Ethereum spec, which from time to time presents challenges. The yellow paper partially serves as a spec. During some periods it was kept up to date wrt protocol changes, during other periods it was not.
- The core devs process is heavily dependent upon the EF and relies on its staff and resources
- Requires that all EIPs be “in the public domain” (not clearly defined). In practice all EIPs are licensed via CC0.

Blockstack current process (https://github.com/blockstack/stacks-blockchain/blob/master/sip/sip-000-stacks-improvement-proposal-process.md)

- Based on BIP2 but vastly simplified
- SIPs currently live the main, core repository. (They should live in their own repository.)
- Workflow is much more implicit than in the above systems (i.e., not explicitly specified).
- Requires that SIPs be reviewed, managed, and accepted by a “core developer committee”, defined as “a group of active contributors to the Stacks blockchain.”
- “Initially the committee will consist of developers from Blockstack PBC. Membership will be opened to the community once mining begins. Nominations to join the committee may be submitted by an existing committee member and approval requires a majority of committee members.”
- A core developer may unilaterally “determine that there is support in the community” (it’s not explained what this means) and assign a SIP number.
- For a soft fork to progress to Final/Active status, “a clear miner majority is required” (also not clear what this means).
- Specifies number of forks: “One hard fork per year will be executed. And additional hard forks will be made in order to fix consensus-critical bugs.” (This does not belong in SIP0 and should be removed.)

Proposal

Like Zcash, the Stacks community should adopt BIP2 with certain relevant revisions (outlined below). High-level “meta governance” questions such as how the Committee is appointed are beyond the scope of this proposal.

Additions and Revisions

- Committee: The Stacks community should appoint (in a manner to be determined) a Technical Steering Committee, akin to Python's. The Committee has ultimate authority over which SIPs are approved, implemented, and adopted.
    - It should consist of a minimum of three members: one appointed by PBC, one by the Foundation, and one by the community (who must be independent of both other organizations).
    - Committee members must be able to demonstrate sufficient technical understanding and ability.
- SIP Editors:
    - Similar to all of the abovementioned communities, a board of several SIP Editors should be established and maintained. This group is at the "front line" for reviewing incoming SIP submissions.
- Types: In addition to the "Types" of BIP outlined in BIP2, Stacks should add several that have worked well for Zcash and Ethereum. A complete list is:
    - Consensus: Affects consensus, i.e., all implementations and the entire network
    - Standard: Other proposals that affect one or more implementations/pieces of code, but not consensus
    - Meta: A proposal specifically about the SIP process itself
    - Process: Any other proposal that does not affect any implementation or code, but does require action on the part of network participants
    - Informational: Any other proposal that does *not* require action on the part of network participants
- Consideration: Each SIP should have one or more "Consideration" (also known colloquially as a "tag") in addition to a single Type. Each Consideration specifies a field or area of expertise that the proposal affects.
    - A partial list of Considerations is: technical, economic, governance, ethics, diversity, community.
    - By default, all proposals have a single "technical" Consideration tag, but a SIP author may choose to remove this tag and/or add others.
    - A subcommittee or advisory council should be established for each of the most important Considerations: e.g., an Economic Council that reviews SIPs tagged with "economic." The subcommittee must review all Accepted SIPs with the relevant Consideration tag, and provide commentary on the impact of this SIP along the lines of this Consideration (e.g., economic impact).
    - Ultimate authority to implement or reject a SIP resides with the Steering Committee, but the Committee should consider the commentary of each subcommittee in the process of making this decision. The Committee may also refer specific questions to the various subcommittees, or request that a particular subcommittee perform additional investigation before making a decision.
- Workflow

    - The process of SIP authorship begins with an informal proposal in an agreed-upon location, such as the Blockstack community forum.
    - Once the proposer determines, at their discretion, that there is a reasonable degree of community acceptance for a given proposal idea, and that it has a reasonable chance of being accepted and implemented, they may author a formal SIP. This author is responsible for gathering the required support for the SIP and for writing the reference implementation (for technical proposals), but they may enlist the help of others, such as core developers, in this process.
    - Any SIP Editor, at their sole discretion, may choose to accept a new SIP, mark it as “Accepted,” assign it a canonical number, and merge it into the canonical SIP repository. Alternatively, they may choose to reject a new SIP on one or more specific grounds, and to notify the author of this decision.
    - The author may respond to a rejection notice and re-submit the SIP in question after concerns have been addressed.
    - The Committee, in its sole discretion, may choose to discuss and vote on any SIP in the “Accepted” state. They may choose to reject the SIP on one or more specific grounds (after which the workflow is as described), or they may choose to accept it and move it to “Recommended” state.
    - Each individual team of core developers may choose to implement any Recommended SIP. It is not explicitly required, but strongly recommended, that all Consensus and Standard track changes made in each individual client implementation flow through the entire SIP process. (This cannot be required since, ultimately, each individual team of core developers is autonomous and accountable only to those who run their software.)
    - Several Recommended SIPs will be bundled together into regular network upgrades. Ultimately, the decision about whether or not to download and upgrade to a specific release of node software is up to each node operator.
    - SIPs that have been included in past network upgrades should be moved to “Implemented.”
    - All statuses: Draft (number unassigned), Accepted, Recommended, Implemented, Rejected, Obsolete, Replaced, Withdrawn.
- Appeals process: Do we want to allow an SIP author to appeal a rejection by an SIP editor, or by the Committee? If so, to whom do they appeal?
- What does and doesn’t need an SIP? Small, implementation-specific details (i.e., those specific to a single client, such as an internal data representation) do not. Anything touching multiple clients, consensus, or other aspects of governance should flow through the SIP process.
- Non-core standards (like Etherem’s ERCs) should be part of a process that’s separate from the SIP process. That process can be much looser, doesn’t necessarily need to involve the same people, etc.

SIP template

- Preamble header (Frontmatter)
    - Title

    - Author
    - Status
    - Type
    - Date created
    - Date updated
    - Backwards compatibility
    - Layer/subprotocols affected
    - Non-technical considerations
    - License
    - Comments URL/Discussions-to
    - Replaces/superseded by (optional)
    - References (optional)
- Abstract
- Specification
- Rationale
- Reference implementation
- Sub-files

Core devs meetings

There should be a quasi-open, regular, recorded meeting of all core Stacks stakeholders.

- "Quasi-open" means that the invitation link to the meeting should be shared openly on a public forum where core stakeholders regularly converse, so that "those who should be there know where to find it."
- Meetings are open to SIP editors and the Committee, as well as anyone with an affiliation with PBC or the Foundation, and to members of the community who have something to contribute.
- From time to time, special guests or subject matter experts may be invited to join meetings to offer commentary on relevant topics.
- The meetings should be recorded and posted to an eternal, public knowledge base (see the current Governance Working Group Calls repository for an example). They may optionally also be livestreamed for public viewing.
- Meetings should be held no less often than monthly. They may be held biweekly or weekly, depending on the pace of development work and the number of issues to discuss.
- There should be a set of no less than two individuals who are trained and prepared to moderate these meetings in an orderly, impartial fashion. They will take turns to moderate the meetings.
- There should additionally be someone assigned to take notes for the meeting. Notes do not need to be a full transcript, but need to cover all major points discussed including decisions and action items.
    - Note that the auto-transcription feature of software like Zoom may be used to expedite this process, but someone still needs to own the task of correcting, finalizing, annotating, and committing the notes.

- Call hosts and note-takers should receive reasonable, predetermined compensation from the Foundation, probably in the form of a fixed, regular bounty.
    - They should also disclose any potential conflicts of interest before participating in meetings.
- Anyone is free to propose items for inclusion on a meeting agenda. In most cases, these items must be encapsulated in a SIP. The agenda for a given meeting is determined at the sole discretion of the moderator for that meeting, but the moderator may not unreasonably refuse to add a relevant agenda item.
- The calls should proceed according to the following agenda:
    - Review of decisions made and action items from prior meeting
    - General update from each client team
    - Testing updates (if distinct from previous bullet point)
    - Review of plans for next network upgrade (hard fork) (if applicable)
        - Timing
        - SIPs slated for inclusion and their statuses
        - Progress towards implementation and testing
    - Discuss agenda items for this call, such as reviewing open/accepted/recommended SIPs
    - Any other items of business
    - Finalize timing of next call
    - Review of decisions made and action items from present meeting