Cassandra Enhancement Proposals (CEP)

This page describes the Cassandra Enhancement Proposal (CEP) process, for putting forward major changes to Cassandra for discussion and decision-making.

Purpose

Cassandra Enhancement Proposals provide a process for the proposal, discussion and endorsement of new feature development in Cassandra. CEPs confer advantages to patch authors by building legitimacy for changes within the community, and obtaining early consent for the direction of development. It is up to feature authors to determine if a CEP should be pursued for any piece of work, balancing the costs of a more burdensome process against the benefits of early community input and endorsement. As CEPs become more common, it is anticipated that project members will become less permissive to large changes that haven't attempted to seek consent beforehand, feeling freer to request major changes that they feel are suitable.

It is highly recommended to pursue a CEP for significant user-facing or changes that cut across multiple subsystems. Community feedback is able to provide information on:

- Unexpected edge cases that may confound work or otherwise complicate it
- Major users and their expectations of existing or future behaviour
- Project attitudes towards specific categories of feature
- Project expectations around how a feature should be structured and delivered

The CEP process aims to be lightweight and flexible. One may be initiated with nothing more than a title to begin, expanding as a working group materialises and ideas crystallise. See Scott Andreas' 2019 NGCC presentation for a community perspective, and how this ties into the lifecycle and evolution of the project.

What a CEP is not

CEPs are not intended to presage a return to waterfall development, and an acceptance of an CEP provides no absolute guarantee that any final product will be accepted. Work-invalidating insights can hit late in development, invalidating an idea, despite significant work being done. The CEP merely mitigates this risk, and helps build legitimacy for a change so that technical difficulties may be discovered early, and non-technical objections handled before work begins.

Who should initiate a CEP?

Anyone can initiate a CEP but you shouldn't do it unless you have the intention and capability to complete the proposed change.

A CEP needs to attract a **Shepherd**, a *Cassandra Committer* committed to guiding the proposal through the process. Although a shepherd may delegate or work with other committers, they are ultimately responsible for the success or failure of a CEP. Responsibilities include, but are not limited to:

- Advocating for the proposal
- Ensuring the working group achieves consensus
- Ensuring the working group seeks feedback from relevant stakeholders and users, and iterates on the design & implementation (see below for additional CEP documentation)
- Ensuring project standards of development and quality are met
- Ensuring changes match the CEP and are absent of critical bugs before releasing them

What should be included in a CEP?

A CEP wiki page should aim to:

- Promote collaboration during the discovery phase of a new feature

- Serve as a permanent document describing the feature that evolves along with its development

A CEP should contain the following sections:

- **Scope**,
- **Goals** (and non-goals),
- **Description of Approach**,
- **Test Plan** covering performance, correctness, failure, and boundary conditions (as applicable),
- **Timeline**,
- **Mail**ing list / **Slack** channels,
- Related **JIRA** tickets.

The Process

Here is the process for making a CEP:

(Optional): For work that is highly fluid and not yet ready for hardening in a wiki article nor broad dev list announcement, create a gdoc with the CEP template and add a link under the "CEP's in draft". This area is recommended for things that are nascent but with a high degree of certainty of eventual dev list proposal of a CEP.

1. To create your own CEP, click on Create CEP.
   If you don't have permission, please send an email with your Wiki ID to `dev@cassandra.apache.org` and request permission. Also add an entry to the table CEPs under discussion.

   Take the next available CEP number and give your proposal a descriptive heading. e.g. "CEP 1: Proposing an Apache Cassandra Management process".

2. Fill in the sections as described above.

3. Start a [DISCUSS] thread on the Apache mailing list. Please ensure that the subject of the thread is of the format *[DISCUSS] CEP-{your CEP number} {your CEP heading}* The discussion should happen on the mailing list not on the wiki since the wiki comment system doesn't work well for larger discussions. In the process of the discussion you may update the proposal. You should let people know the changes you are making.

4. As the CEP nears completion, consider adding any additional design documentation (see below) to the CEP, especially where it summaries working group discussions.

5. Once the proposal is finalized call a [VOTE] to have the proposal adopted. These proposals are more serious than code changes and more serious even than release votes. The criteria for acceptance is consensus (3 binding +1 votes and no binding vetoes). The vote should remain open for 72 hours.

6. ***Please*** update the CEP wiki page, and the index below, to reflect the current stage of the CEP after a vote. This acts as the permanent record indicating the result of the CEP (e.g., Accepted or Rejected). Also report the result of the CEP vote to the voting thread on the mailing list so the conclusion is clear.

Example CEP Design Documentation

After the CEP is opened and a working group is active, to help flesh out the implementation constraints, here are some suggestions for additional discussion and documentation that can go into the CEP:

- **Motivation**: The problem to be solved.
- **Audience**: The intended client audience. Examples include data scientists, data engineers, library devs, devops, etc. A single CEP can have multiple target personas.
- **Proposed Change**: The new thing you want to do. This may be fairly extensive and have large subsections of its own. Or it may be a few sentences, depending on the scope of the change.
- **New or Changed Public Interfaces**: Impact to any of the "compatibility commitments" described above. We want to call these out in particular so everyone thinks about them.
- **Migration Plan and Compatibility**: If this feature requires additional support for a no-downtime upgrade describe how that will work.
- **Rejected Alternatives**: What are the other alternatives you considered and why are they worse? The goal of this section is to help people understand why this is the best solution now, and also to prevent churn in the future when old alternatives are reconsidered.

Compatibility Concerns

Cassandra requires a high level of compatibility between releases to ensure rolling upgrades are possible, as well as supporting third-party libraries and tools.

These areas of compatibility are

- native protocol (and CQL)

- gossip and the messaging service
- pluggable components (SPIs) like authorisation, triggers, …
- commitlog, hintlog, cache files
- sstables components
- configuration
- jmx mbeans (including metrics)
- monitoring
- client tool classes
- command line tools and arguments
- operational routines
- …

List of CEPs

Adopted CEPs

CEP	Release
CEP-3: Guardrails	4.1
CEP-7: Storage Attached Index	5.0
CEP-8: Drivers Donation	
CEP-9: Make SSLContext creation pluggable	4.1
CEP-10: Cluster and Code Simulation	4.1
CEP-11: Pluggable memtable implementations	4.1
CEP-13: Denylisting partitions	4.1
CEP-14: Paxos Improvements	4.1
CEP-15: General Purpose Transactions	
CEP-16: Auth Plugin Support for CQLSH	4.1
CEP-17: SSTable format API	5.0
CEP-19: Trie memtable implementations	5.0
CEP-20: Dynamic Data Masking	5.0
CEP-21: Transactional Cluster Metadata	
CEP-25: Trie Indexed SSTable	5.0
CEP-26: Unified Compaction Strategy	5.0
CEP-28: Reading and Writing Cassandra Data with Spark Bulk Analytics	5.0
CEP-29: CQL NOT operator	
CEP-30: Approximate Nearest Neighbor(ANN) Vector Search via Storage-Attached Indexes	
CEP-33: CIDR filtering authorizer	5.0

CEPs under discussion

CEP	Comment
CEP-1: Apache Cassandra Management Process(es)	Sent emails to Dev discussion group.
CEP-2: Kubernetes Operator	Emails periodically sent to dev list. SIG meetings held periodically.
CEP-12: Diagnostic Events in virtual tables	Sent emails to Dev discussion group.

CEP-24: Password validation and generation	Sent emails to Dev discussion group.
CEP-40: Data Transfer Using Cassandra Sidecar for Live Migrating Instances	Sent email to Dev discussion group.

CEPs in draft

gdoc link	Comment
CEP-4: EXPLAIN	Not discussed yet
CEP-5: JOINs (copy of gdoc w/permissions changed)	Not discussed yet
CEP-6: Change Data Capture v2	Not discussed yet
CEP-27: Generic API for Internal Data Collections Exposure	dev-list: Discussion

Dormant / Inactive CEPs

CEP	Comment

Discarded CEPs

CEP	Comment
CEP-18: Improving Modularity	CEP withdrawn, discussion ended up with wanted to consider each ticket on their own, rather than considering them as a whole in a CEP. See email thread on dev@ discussion group.
CEP-23: Enhancement for Sparse Data Serialization	CEP withdrawn, discussion indicated that CEP was not the proper form for the change. Change can not be made within the confines of the the outlined CEP.