Samza Enhancement Proposal

This page describes a proposed Samza Enhancement Proposal (SEP) process for proposing a major change to Apache Samza.

To create a SEP, click on "Create" at the top of Header and select "SEP Template" from the dialog to create a new page.

Why SEP ?

The purpose of SEP is to collate and document all planned major enhancement to Apache Samza. JIRA is still the tool for tracking bugs and progress, SEP give an accessible overview of the design proposal, discussions and final acceptance status of the proposal. SEP, in other words, is nothing but a central location for all design documents in Apache Samza.

We want to make Samza a core architectural component for users. We also support a large number of integrations with other systems. Keeping this kind of usage healthy requires a high level of compatibility between releases — core architectural elements can't break compatibility or shift functionality from release to release. As a result each new major feature or public API has to be done in a way that we can stick with it going forward.

This means when making this kind of change we need to think through what we are doing as best we can prior to release. And as we go forward we need to stick to our decisions as much as possible. All technical decisions have pros and cons so it is important we capture the thought process that lead to a decision or design to avoid flip-flopping needlessly.

Hopefully we can make these proportional in effort to their magnitude — small changes should just need a couple brief paragraphs, whereas large changes need detailed design discussions. This process also isn't meant to discourage incompatible changes — proposing an incompatible change is totally legitimate. Sometimes we will have made a mistake and the best path forward is a clean break that cleans things up and gives us a good foundation going forward. Rather this is intended to avoid accidentally introducing half thought-out interfaces and protocols that cause needless heartburn when changed. Likewise the definition of "compatible" is itself squishy: small details like which errors are thrown when are clearly part of the contract but may need to change in some circumstances, likewise performance isn't part of the public contract but dramatic changes may break use cases. So we just need to use good judgement about how big the impact of an incompatibility will be and how big the payoff is.

When to provide a SEP?

Any of the following should be considered a major change:

- Any major new feature, subsystem, or piece of functionality
- Any change that impacts the public interfaces of the project

The following are treated as public interfaces of Samza

- Any class in the public package - samza-api
- Any configuration change in Samza

Any major **functional** change to Samza's core functionality, that is changes to the behavior of the classes in samza-core package also requires a SEP.

Not all compatibility commitments are the same. We need to spend significantly more time on public APIs and samza-core api behavior as these break code for users. They cause people to rebuild code and lead to compatibility issues in large multi-dependency projects (which end up requiring multiple incompatible versions). Configuration, monitoring, and command line tools can be faster and looser — changes here will break monitoring dashboards and require a bit of care during upgrades but aren't a huge burden. For the most part monitoring, command line tool changes, and configs are added with new features so these can be done with a single SEP.

What should be included in SEP?

A typical SEP should include the following sections:

- **Problem**: Describe the problem to be solved
- **Motivation**: Why the problem should be solved
- **Proposed Changes**: Describe the changes 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, if any**: Impact to any of the "compatibility commitments" described above. We want to call these out in particular so everyone thinks about them.
- **Implementation / Test Plan** (if applicable): Some large projects may consists of multiple sub-projects and requires more planning for implementation and testing. It may be prudent to call them out in the proposal.
- **Migration Plan and Compatibility**: If this feature requires additional support for a no-downtime upgrade describe how that will work. If a no-downtime upgrade for users cannot be support, describe the migration plan for the users.
- **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.

Who should initiate and drive a SEP?

Anyone can initiate a SEP but you shouldn't do it unless you have an intention of getting the work done to implement it (otherwise it is silly).

What is the criteria to accept / reject a SEP?

We will be following the Apache voting guideline to drive our decisions regarding an SEP. Decisions will be made on the [VOTE] mailing thread for a particular SEP. All community members are encouraged to participate by replying to the VOTE mailing thread. The votes can have the following meaning:

Vote	Meaning
+1	'Yes,' 'Agree,' or 'the action should be performed.'
0	Neutral about the proposed action (or mildly negative but not enough so to want to block it
-1	This is a negative vote. On issues where consensus is required, this vote counts as a **veto**. All vetoes must contain an explanation of why the veto is appropriate. Vetoes with no explanation are void. It may also be appropriate for a -1 vote to include an alternative course of action.

The criteria for acceptance of a SEP is lazy majority.

Process

Here is the process for making a SEP:
1. Create a page which is a child of this one. Take the next available SEP number and give your proposal a descriptive heading. e.g. "SEP 73: Samza Standalone with ZK Coordination". If you don't have the necessary permissions for creating a new page, please ask on the dev mailing list.
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] SEP-{your SEP number} {your SEP heading} The discussion should happen on the mailing list and *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. 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 lazy majority.
5. **Please** update the SEP wiki page, and the index below, to reflect the current stage of the SEP after a vote. This acts as the permanent record indicating the result of the SEP (e.g., Accepted or Rejected). Also report the result of the SEP vote to the voting thread on the mailing list so the conclusion is clear.

SEP Number

Next SEP Number: 33

Use this number as the identifier for your SEP and increment this value.

SEP Index

SEP	Status	Link to Discussion Thread	Related JIRA
SEP-1: Semantics of ProcessorId in Samza	Accepted	http://mail-archives.apache.org/mod_mbox/samza-dev/201703.mbox/browser	SAMZA-1126
SEP-2: ApplicationRunner Design	Discuss		SAMZA-1130
SEP-3: Heart-beat mechanism between JobCoordinator and all running containers	Accepted	http://mail-archives.apache.org/mod_mbox/samza-dev/201705.mbox/%3CCANxwKLaVro6MBvUJW2RvoNLDO9-G87Y3Ox%2B5W66K_CxBqeVfgQ%40mail.gmail.com%3E	SAMZA-871
SEP-4: Adjunct Data Store for Unbounded Datasets	Discuss	http://mail-archives.apache.org/mod_mbox/samza-dev/201705.mbox/browser	SAMZA-1278
SEP-5: Enable partition expansion of input streams	Accepted		SAMZA-1293
SEP-6: Support Control Message Across Intermediate Streams	Discuss		SAMZA-1260
SEP-7: Samza on Azure	Discuss		SAMZA-1373
SEP-8 Add in-memory system consumer & producer	Accepted	http://mail-archives.apache.org/mod_mbox/samza-dev/201708.mbox/%3CCAG2vMJHyxCKqn4K+pst83OCpjc1r79MAPsqY-VnKBbPUBUEc0g@mail.gmail.com%3E	SAMZA-1395
SEP-9 Add a Kinesis SystemConsumer and SystemProducer			
SEP-10 Exactly-once Processing in Samza			
SEP-11: Host affinity in standalone.	Accepted		SAMZA-1554

SEP-12: Integration Test Framework	Accepted	http://mail-archives.apache.org/mod_mbox/samza-dev/201805.mbox/%3CDM5PR21MB02827A6FA9F47CB8EF99A339A2810%40DM5PR21MB0282.namprd21.prod.outlook.com%3E	SAMZA-1629
SEP-13: unify high- and low-level user applications in YARN and standalone	Accepted	https://lists.apache.org/thread.html/5be324d239633f7433525b0e52f0377ad2f8c25787eefba1b96a492c@%3Cdev.samza.apache.org%3E	SAMZA-1789
SEP-14: System and Stream Descriptors	Accepted		SAMZA-1804
SEP-15: New Runtime Context API	Accepted		SAMZA-1714
SEP-16: Extend ExecutionPlanner to Support Stream-Table Join	Accepted		SAMZA-1889
SEP-17: Samza SQL Shell	Accepted		
SEP-18: Startpoints - Manipulating Starting Offsets for Input Streams	Accepted		SAMZA-1983
SEP-19: Hot standby state for Samza applications	Discuss		
SEP-20: Samza on Kubernetes	Accepted		
SEP-21: Samza Async API for High Level	Accepted		SAMZA-2055
SEP-22: Container Placements in Samza	Accepted	http://mail-archives.apache.org/mod_mbox/samza-dev/202001.mbox/%3CCAKkRg%3D94NY8cLn89u%3DVeL1K52R3XuOimzxXsy7BLzS7fpS%3DLfg%40mail.gmail.com%3E	SAMZA-2373
SEP-23: Simplify Job Runner	Accepted	https://mail-archives.apache.org/mod_mbox/samza-dev/201912.mbox/%3C560AD519-A166-4024-B1D6-E1DAE44611A4%40gmail.com%3E	SAMZA-2405
SEP-24: Cluster-based Job Coordinator Dependency Isolation	Accepted	https://mail-archives.apache.org/mod_mbox/samza-dev/202003.mbox/%3CCABbqq3yEdAXiiTzXtO%2B%3DUnCBvxFR-5q_aJx3uqcQOoxF7M09vQ%40mail.gmail.com%3E	
SEP-25: PR Title And Description Guidelines	Accepted	http://mail-archives.apache.org/mod_mbox/samza-dev/201912.mbox/%3CCAMja7KeQr9C048UVZwfSC46h%3DEX_9S%2BSEvMF9NPg0V5dPTPfZg%40mail.gmail.com%3E	
SEP-26: Azure Blob Storage Producer	Accepted	https://lists.apache.org/thread/sdopjojgcchrgmphfco98hbm5posbm9c	SAMZA-2421
SEP-27: Side Inputs for Local Stores	Discuss		SAMZA-1773
SEP-28: Samza State Backend Interface and Checkpointing Improvements	Accepted	https://mail-archives.apache.org/mod_mbox/samza-dev/202106.mbox/%3CCA%2B6YmWVVxz%3DXr244rPG2a-a6QAoR0mjrW9CK41-U7tSuv8oY4Q%40mail.gmail.com%3E	SAMZA-2591
SEP-29: Blob Store Based State Backup And Restore	Accepted	https://mail-archives.apache.org/mod_mbox/samza-dev/202106.mbox/%3CCAMja7KcPM4c8iQwMu1wC3tkFmtzUf6U6UK2LKWGbiiucHjDw3w%40mail.gmail.com%3E	SAMZA-2657
SEP-30: Suport Partial Updates in Table API	Accepted	https://www.mail-archive.com/dev@samza.apache.org/msg09162.html	SAMZA-2709
SEP-31: Pipeline Drain	Discuss	https://lists.apache.org/thread/7m2hqcqq9lx9o1d48gb64glplb3g2crt	SAMZA-2741
SEP-32: Elasticity for samza	Discuss		SAMZA-2687