CommitPolicy

Commit Policy

This document is describes how Solr development has been done in the past, to help guide how we as a community may choose to do development in the future.

Please Note: These guidelines are for Committers making changes directly to the revision control system. Community members wishing to contribute patches should not feel burdened by any policies spelled out on this page, instead please consult HowToContribute for more information.

People may want to refer to the Apache Glossary for clarification on the concepts of **CTR** (Commit-Then-Review) Policies and **RTC** (Review-Then-Commit) Policies – but please keep in mind that the Solr community (like our Lucene parent community) is very relaxed, so even in cases where we talk about "Review Then Commit" we tend to be extremely informal, and rarely (if ever) take votes as part of the "Review" process.

There is a lot of trust between committers, in some cases it may make sense to diverge from these policies (more info at the end of this page).

See also the dev list discussion of 2007-04-27 about when to commit what.

Trivial Bug Fixes, Typo corrections, and Adding Documentation

Committers should feel free to makes simple changes at will, under the **CTR** model.

Bug Fix commits should always include at least one Unit Test that does not pass without the accompanying fix.

Adding New Unit Tests of Existing Functionality

Committers should feel free to commit additional unit tests at will to help clarify and future proof existing behavior under the **CTR** model.

Adding New Functionality

New functionality should preferably first be submitted as a patch in the issue tracking system under the **RTC** model since adding additional functionality invariably involves adding new APIs. Rolling back API additions is non-trivial, so it's good to get feedback first.

If no one provides feedback on a feature after some "reasonable" amount of time (relative to the scope of the new feature), a committer who is comfortable with the API of a new feature should feel to commit without review and rely on the **CTR** model as a fallback.

No new Functionality should ever be committed without some Unit Tests demonstrating successful behavior, and ideally a few Unit Tests demonstrating that the feature "fails correctly" on invalid input.

All new Features should be accompanied by new documentation (at a minimum, mention on the wiki)

Architectural or "Weighty" Changes to the "Guts" of Solr

Changes to Solr's internals should always be tracked as a patch in the issue tracking system under the **RTC** model. At least one other person knowledgeable in the code that is changing should review the patch before it is committed.

Changes of this scope should almost always include new Unit Tests of whatever internal APIs are changing.

if an architectural or other serious internal change requires changes to existing unit tests, that should be pointed out and the potential ramifications for existing end users should be discussed thoroughly on the Solr developer list.

Non-Backwards Compatible Changes to Functionality or APIs

Changes to Solr's public APIs (either via URL structure, or APIs exposed to Plugin Developers) should always be tracked as a patch in the issue tracking system under the **RTC** model. Potential ramifications for existing end users should be discussed thoroughly on the Solr user list.

Changes of this scope should result in a Major Version increase (ie: 1.x to 2.0)

Deviating From These Guidelines

...then, suddenly, everybody went silent in the room...

What, deviating? The page title says POLICY, not?

At the scale of this project it seems easier to take into account the amount of trust that exists between community members, as opposed to creating zillions of rules.

To paraphrase Yonik: *a half-baked patch submitted to JIRA with no documentation, no tests and no backwards compatibility...is better than no patch at all!*

So we won't create a policy about when it makes sense to deviate from these guidelines....but in some cases it does. In case of doubt, just ask on the dev list.