

---

# Proposing Changes to Go

## Introduction

The Go project's development process is design-driven. Significant changes to the language, libraries, or tools (which includes API changes in the main repo and all [golang.org/x](https://golang.org/x) repos, as well as command-line changes to the `go` command) must be first discussed, and sometimes formally documented, before they can be implemented.

This document describes the process for proposing, documenting, and implementing changes to the Go project.

To learn more about Go's origins and development process, see the talks [How Go Was Made](#), [The Evolution of Go](#), and [Go, Open Source, Community](#) from GopherCon 2015.

## The Proposal Process

The proposal process is the process for reviewing a proposal and reaching a decision about whether to accept or decline the proposal.

1. The proposal author creates a brief issue describing the proposal.  
Note: There is no need for a design document at this point.  
Note: A non-proposal issue can be turned into a proposal by simply adding the proposal label.  
Note: Language changes should follow a separate template
2. A discussion on the issue tracker aims to triage the proposal into one of three outcomes:
  - Accept proposal, or
  - decline proposal, or
  - ask for a design doc.

If the proposal is accepted or declined, the process is done. Otherwise the discussion is expected to identify concerns that should be addressed in a more detailed design.

3. The proposal author writes a design doc to work out details of the proposed design and address the concerns raised in the initial discussion.
4. Once comments and revisions on the design doc wind down, there is a final discussion on the issue, to reach one of two outcomes:
  - Accept proposal or
  - decline proposal.

---

After the proposal is accepted or declined (whether after step 2 or step 4), implementation work proceeds in the same way as any other contribution.

## Detail

### Goals

- Make sure that proposals get a proper, fair, timely, recorded evaluation with a clear answer.
- Make past proposals easy to find, to avoid duplicated effort.
- If a design doc is needed, make sure contributors know how to write a good one.

### Definitions

- A **proposal** is a suggestion filed as a GitHub issue, identified by having the Proposal label.
- A **design doc** is the expanded form of a proposal, written when the proposal needs more careful explanation and consideration.

### Scope

The proposal process should be used for any notable change or addition to the language, libraries and tools. “Notable” includes (but is not limited to):

- API changes in the main repo and all [golang.org/x](https://golang.org/x) repos.
- Command-line changes to the [go](https://golang.org/cmd/go) command.
- Any visible behavior changes that need a GODEBUG setting for compatibility.
- Any other visible behavior changes in existing functionality.
- Adoption or use of new protocols, protocol versions, cryptographic algorithms, and the like, even in an implementation. Such changes are externally visible and require discussion and probably a GODEBUG setting.

Since proposals begin (and will often end) with the filing of an issue, even small changes can go through the proposal process if appropriate. Deciding what is appropriate is matter of judgment we will refine through experience. If in doubt, file a proposal.

There is a short list of changes that are typically not in scope for the proposal process:

- Making API changes in internal packages, since those APIs are not publicly visible.
- Making API or command-line changes in [golang.org/x/build](https://golang.org/x/build), since that is code to run the Go project, not for users to import and depend on.

- 
- Adding new system call numbers or direct system call wrappers (`//sys` lines) in [golang.org/x/sys](https://golang.org/x/sys).
  - Adding new C-equivalent data structures to support those system calls.

Again, if in doubt, file a proposal.

## Compatibility

Programs written for Go version 1.x must continue to compile and work with future versions of Go 1. The Go 1 compatibility document describes the promise we have made to Go users for the future of Go 1.x. Any proposed change must not break this promise.

## Language changes

In 2018 we started a Go 2 process during which we may change the language, as described on the Go blog. Language changes should follow the proposal process described here. As explained in the blog entry, language change proposals should

- address an important issue for many people,
- have minimal impact on everybody else, and
- come with a clear and well-understood solution.

Proposals should follow the Go 2 template. See the Go 2 review minutes and the release notes for examples of recent language changes.

## Design Documents

As noted above, some (but not all) proposals need to be elaborated in a design document.

- The design doc should be checked in to the proposal repository as `design/NNNN-shortname.md`, where `NNNN` is the GitHub issue number and `shortname` is a short name (a few dash-separated words at most). Clone this repository with `git clone https://go.googlesource.com/proposal` and follow the usual Gerrit workflow for Go.
- The design doc should follow the template.
- The design doc should address any specific concerns raised during the initial discussion.
- It is expected that the design doc may go through multiple checked-in revisions. New design doc authors may be paired with a design doc “shepherd” to help work on the doc.

- 
- For ease of review with Gerrit, design documents should be wrapped around the 80 column mark. Each sentence should start on a new line so that comments can be made accurately and the diff kept shorter.
    - In Emacs, loading `fill.el` from this directory will make `fill-paragraph` format text this way.
  - Comments on Gerrit CLs should be restricted to grammar, spelling, or procedural errors related to the preparation of the proposal itself. All other comments should be addressed to the related GitHub issue.

### **Quick Start for Experienced Committers**

Experienced committers who are certain that a design doc will be required for a particular proposal can skip steps 1 and 2 and include the design doc with the initial issue.

In the worst case, skipping these steps only leads to an unnecessary design doc.

### **Proposal Review**

A group of Go team members holds “proposal review meetings” approximately weekly to review pending proposals.

The principal goal of the review meeting is to make sure that proposals are receiving attention from the right people, by cc’ing relevant developers, raising important questions, pinging lapsed discussions, and generally trying to guide discussion toward agreement about the outcome. The discussion itself is expected to happen on the issue tracker, so that anyone can take part.

The proposal review meetings also identify issues where consensus has been reached and the process can be advanced to the next step (by marking the proposal accepted or declined or by asking for a design doc).

Minutes are posted to [golang.org/s/proposal-minutes](https://golang.org/s/proposal-minutes) after the conclusion of the weekly meeting, so that anyone interested in which proposals are under active consideration can follow that issue.

Proposal issues are tracked in the Proposals project on the Go issue tracker. The current state of the proposal is captured by the columns in that project, as described below.

The proposal review group can, at their discretion, make exceptions for proposals that need not go through all the stages, fast-tracking them to Likely Accept/Likely Decline or even Accept/Decline, such as for proposals that do not merit the full review or that need to be considered quickly due to pending releases.

---

**Incoming** New proposals are added to the Incoming column.

The weekly proposal review meetings aim to look at all the issues in the Active, Likely Accept, and Likely Decline columns. If time is left over, then proposals from Incoming are selected to be moved to Active.

Holding proposals in Incoming until attention can be devoted to them (at which they move to Active, and then onward) ensures that progress is made moving active proposals out to Accepted or Declined, so we avoid receive livelock, in which accepting new work prevents finishing old work.

**Active** Issues in the Active column are reviewed at each weekly proposal meeting to watch for emerging consensus in the discussions. The proposal review group may also comment, make suggestions, ask clarifying questions, and try to restate the proposals to make sure everyone agrees about what exactly is being discussed.

**Likely Accept** If an issue discussion seems to have reached a consensus to accept the proposal, the proposal review group moves the issue to the Likely Accept column and posts a comment noting that change. This marks the final period for comments that might change the recognition of consensus.

**Likely Decline** If a proposal discussion seems to have reached a consensus to decline the proposal, the proposal review group moves the issue to the Likely Decline column. An issue may also be moved to Likely Decline if the proposal review group identifies that no consensus is likely to be reached and that the default of not accepting the proposal is appropriate.

Just as for Likely Accept, the group posts a comment noting the change, and this marks the final period for comments that might change the recognition of consensus.

**Accepted** A week after a proposal moves to Likely Accept, absent a change in consensus, the proposal review group moves the proposal to the Accepted column. If significant discussion happens during that week, the proposal review group may leave the proposal in Likely Accept for another week or even move the proposal back to Active.

Once a proposal is marked Accepted, the Proposal-Accepted label is applied, it is moved out of the Proposal milestone into a work milestone, and the issue is repurposed to track the work of implementing the proposal.

The default work milestone is Backlog, indicating that the work applies to the Go release itself but is not slated for a particular release. Another common next milestone is Unreleased, used for work that is not part of any Go release (for example, work in parts of [golang.org/x](https://golang.org/x) that are not vendored into the standard releases).

---

**Declined** A week after a proposal moves to Likely Decline, absent a change in consensus, the proposal review group moves the proposal to the Declined column. If significant discussion happens during that week, the proposal review group may leave the proposal in Likely Decline for another week or even move the proposal back to Active. Once a proposal is marked Declined, it is closed.

**Declined as Duplicate** If a proposal duplicates a previously decided proposal, the proposal review group may decline the proposal as a duplicate without progressing through the Active or Likely Decline stages.

Generally speaking, our approach to reconsidering previously decided proposals follows John Ousterhout's advice in his post "Open Decision-Making," in particular the "Reconsideration" section.

**Declined as Infeasible** If a proposal directly contradicts the core design of the language or of a package, or if a proposal is impossible to implement efficiently or at all, the proposal review group may decline the proposal as infeasible without progressing through the Active or Likely Decline stages.

If it seems like there is still general interest from others, or that discussion may lead to a feasible proposal, the proposal may also be kept open and the discussion continued.

**Declined as Retracted** If a proposal is closed or retracted in a comment by the original author, the proposal review group may decline the proposal as retracted without progressing through the Active or Likely Decline stages.

If it seems like there is still general interest from others, the proposal may also be kept open and the discussion continued.

**Declined as Obsolete** If a proposal is obsoleted by changes to Go that have been made since the proposal was filed, the proposal review group may decline the proposal as obsolete without progressing through the Active or Likely Decline stages.

If it seems like there is still general interest from others, or that discussion may lead to a different, non-obsolete proposal, the proposal may also be kept open and the discussion continued.

**Hold** If discussion of a proposal requires design revisions or additional information that will not be available for a couple weeks or more, the proposal review group moves the proposal to the Hold column with a note of what it is waiting on. Once that thing is ready, anyone who can edit the issue tracker can move the proposal back to the Active column for consideration at the next proposal review meeting.

---

## Consensus and Disagreement

The goal of the proposal process is to reach general consensus about the outcome in a timely manner.

If proposal review cannot identify a general consensus in the discussion of the issue on the issue tracker, the usual result is that the proposal is declined. It can happen that proposal review may not identify a general consensus and yet it is clear that the proposal should not be outright declined. As one example, there may be a consensus that some solution to a problem is important, but no consensus on which of two competing solutions should be adopted.

If the proposal review group cannot identify a consensus nor a next step for the proposal, the decision about the path forward passes to the Go architects (currently gri@, iant@, and rsc@), who review the discussion and aim to reach a consensus among themselves. If so, they document the decision and its rationale on the issue.

If consensus among the architects cannot be reached, which is even more unusual, the arbiter (currently rsc@) reviews the discussion and decides the next step, documenting the decision and its rationale on the issue.

## Help

If you need help with this process, please contact the Go contributors by posting to the golang-dev mailing list. (Note that the list is moderated, and that first-time posters should expect a delay while their message is held for moderation.)

To learn about contributing to Go in general, see the contribution guidelines.