A RESOURCE FOR DESIGN SYSTEM TEAMS

One-page RFC template.

For changes other teams will have to live with.

One page. Five sections. A dated decision.

24	05	400
items, each with its evidence	sections, problem to decision	words: the page limit the linter holds

Filter at the proposal, not at the revert.

Someone ships a "quick improvement" without review, three teams copy it, and by the time anyone notices, reverting costs more than fixing forward. The essay behind this guide saw those RFC-less changes create three-week cleanup cycles at SAP Build Apps. Its fix is a gate that costs one page: problem, proposal, impact, alternatives, decision.

Rust, React and Carbon ask for an RFC only for "substantial" changes [S01, S02, S03]. Architecture decision records run one or two pages [S05]; Google's mini design docs one to three [S06]. This guide takes the shortest form and makes the limit and the decision trail checkable.

The kit's linter needs no packages. It reports no errors on RFC-0042, which adds `LinkButton` to the Acme UI example system, and 11 on RFC-0043, a well-meant proposal that skips every step. RFC-0042 ships on the October 2026 train from Field Guide 08 and registers deprecation DEP-0007 from Field Guide 09.

Version 1.0 / Sources checked 24 September 2026
Field Guide 07 of the Design × AI series, with 08 and 09 from the essay Enterprise UX that ships. Kit verified 24 September 2026 on Node 22.22; it needs Node 20 or later.

Practical guidance, not a standard. The five sections follow the essay's template; borrowed ideas are credited where used. Acme UI, its people and its issues are a worked example, not a real system. Prepared with AI assistance and edited by hand.

START HERE

Suggest, propose, implement.

The essay's contribution model has three tiers. Most ideas stay at the first, and that is the system working. An RFC is the second tier: the point where an idea has to justify its cost to everyone who will maintain it.

The change	Route	Why
Bug fix, docs fix, refactor, test	Pull request	Shape changes, meaning does not [S01].
New value in an existing set	Issue; a maintainer decides	Cheap to reverse, no new concept.
New component, public prop or token category	RFC	API every consumer inherits [S02].
Deprecation or breaking change	RFC plus a DEP entry (Guide 09)	Other teams pay for the migration.
Process change, including this template	RFC	Everyone plans around it.

Put your version in CONTRIBUTING.md and link to it when you redirect a pull request or an issue. Owners: maintainers run Sections 01 and 04 with the deciders; authors own Sections 02 and 03.

Writing your first RFC

Copy `rfc-template.md`, read Section 02, run `lint-rfc.mjs` until it is clean. No evidence for Problem? Open an issue instead.

Reviewing one

Section 04 and P24. The linter checks the shape; you check the evidence, the alternatives and the costs.

Running the process

Sections 01 and 04: what needs an RFC, who decides, how long comments stay open, which train ships it.

Adding it to CI

Appendix B. Any error exits non-zero, so an RFC cannot merge in a state the process does not allow.

Label	Meaning
`TEMPLATE`	A section or prompt in `rfc-template.md`.
`LINT`	Enforced by `lint-rfc.mjs`; the evidence line names the rule code.
`SCHEMA`	A field enforced by `rfc-frontmatter.schema.json` (rule F1).
`PROCESS`	A step in the review that leaves a record: a link, a date, a status.
`PRACTICE`	A working method with a review signal rather than a gate.

SECTION 01

Decide what needs an RFC.

An RFC process fails in two directions: everything needs one and nobody contributes, or nothing does and the system fragments. Draw the line once, in writing.

- [ ] **Write down which changes need an RFC**

  P01 PROCESS

  New public API, removal of shipped API, new conventions and process changes. Everything else is an issue or a pull request. Rust, React and Carbon all define the gate as "substantial" and list what does not count.

  Evidence: The route table is in CONTRIBUTING.md, and redirects in review link to it.

  S01, S02, S03.

- [ ] **Start at Suggest: an issue with no template**

  P02 PROCESS

  Tier one costs nothing: describe the need. Maintainers answer with "write an RFC", "we will do it" or "not in the system, and why". The essay estimates that about 20% of suggestions reach the RFC stage.

  Evidence: Every RFC's Problem section links the issue it grew from (rule E1 requires a link).

  The essay's three-tier model; Nathan Curtis separates small contributions from large ones that go through propose, design, code, docs and release. S08.

- [ ] **The RFC comes before the code**

  P03 PROCESS

  Tier three is implementation, after approval. A pull request that implements an undecided RFC waits, however good it is.

  Evidence: The implementation pull request links an RFC whose status is `accepted`.

  Essay; Rust implements only "active" RFCs. S01.

- [ ] **Withdrawal is a result, not a failure**

  P04 PRACTICE

  Many requests withdraw themselves once the author works through the template. Record that with status `withdrawn`, so the next person with the same idea finds the reasoning.

  Evidence: Withdrawn RFCs stay in the index with their Decision section.

  Essay; Oxide keeps abandoned RFDs as a state. S04.

Most ideas should stop at Suggest. That is the filter working.

SECTION 02

Fit the argument on one page.

One page forces clarity; longer documents hide weak thinking in verbosity. That is the essay's claim, and the linter turns it into a number.

- [ ] **One page, measured**

  P05 LINT

  At most 400 words and 45 non-blank lines, template comments excluded. Rendered at 11 pt on A4, 400 words under the five headings fill 0.89 of a page; the 341-word worked example, with a code block and lists, fills 0.90. Put detail behind links.

  Evidence: Rules L1 and L2. RFC-0043 fails L1 at 578 words.

  Essay: "One page forces clarity." ADRs: one or two pages; mini design docs: one to three. S05, S06.

- [ ] **Five sections, in order, and nothing else**

  P06 TEMPLATE LINT

  Problem, Proposal, Impact, Alternatives considered, Decision. One optional section, Open questions. A "Background" section is how an RFC grows a second page.

  Evidence: Rule S1 (missing, unknown or out of order) and S2 (empty after comments are removed).

  The essay's template. Rust and React split the same content into nine sections. S02, S11.

- [ ] **Problem carries evidence**

  P07 LINT

  What goes wrong, for whom, how often, with links: bug reports, research notes, a usage count. "Stakeholders feel it is flat" is an opinion. No evidence, no RFC: open an issue instead.

  Evidence: Rule E1: the Problem section must contain a link or an issue reference.

  Essay: "Include evidence." Rust: RFCs without convincing motivation are poorly received. S01.

- [ ] **Proposal shows the code a consumer will write**

  P08 TEMPLATE

  The API surface, one snippet, and what is out of scope. A reviewer who can see `<LinkButton href>` can argue with it; a paragraph about "navigation affordances" cannot be reviewed.

  Evidence: A fenced code block in Proposal; reviewers ask for one when it is missing.

  React's template leads with a basic example; Google design docs state goals and non-goals. S02, S06.

- [ ] **Impact answers four questions**

  P09 LINT

  Teams affected, with the count you measured. Breaking change: yes or no. Migration: yes or no, and how. Costs and drawbacks: what gets worse.

  Evidence: Rule I1: both yes/no lines present, and "Breaking change" agrees with `breaking` in the frontmatter.

  Essay's Impact section; Rust and React ask for drawbacks explicitly. S01, S11.

- [ ] **Alternatives include doing nothing**

  P10 LINT

  At least two options, one of them "Do nothing" with its consequence. If doing nothing is fine, the RFC is not needed.

  Evidence: Rule A1.

  Rust "Rationale and alternatives"; MADR "Considered Options"; Google "Alternatives considered". S06, S07, S11.

- [ ] **The decision is a sentence with a date**

  P11 LINT

  Accepted, Accepted with changes (list them), Deferred until a date, or Rejected, each with the reason. Until then the section says "Pending".

  Evidence: Rule D1: the section's first word matches `status`, and a decided RFC carries a `decided` date.

  Essay's four outcomes; Nygard records status and consequences. S05.

- [ ] **No template residue**

  P12 LINT

  Placeholder text left in (`TBD`, `Your Name`, `RFC-0000`) means nobody read the page before asking others to.

  Evidence: Rule P1, with the placeholder list in `rfc-policy.json`.

  Kit rule.

If it does not fit on a page, it is two decisions or none.

SECTION 03

Make the status machine-readable.

The frontmatter is what an index, a release calendar, a deprecation registry and an agent read. Keep it small and closed.

- [ ] **Frontmatter validates against the schema**

  P13 SCHEMA

  `id`, `title`, `status`, `authors`, `deciders`, `created`, `discussion`, `commentsClose`, `packages`, `breaking`, `deprecations`; `decided`, `target` and `supersededBy` when they apply. Unknown fields fail.

  Evidence: Rule F1 against `rfc-frontmatter.schema.json` (JSON Schema 2020-12).

  Oxide RFDs carry authors, state, discussion link and labels; MADR carries status, date and decision-makers. S04, S07.

- [ ] **Status comes from a fixed set**

  P14 SCHEMA

  `draft`, `discussion`, `accepted`, `deferred`, `rejected`, `withdrawn`, `superseded`. Anything else is a status nobody can filter on.

  Evidence: The `status` enum.

  Maps to Rust's active, postponed and closed, Oxide's six states and Nygard's proposed, accepted, deprecated and superseded. S01, S04, S05.

- [ ] **Name the deciders**

  P15 SCHEMA PROCESS

  A person or a named group decides. Not "everyone who commented", and not the author. Comments inform; deciders decide.

  Evidence: `deciders` is required, and the Decision section names them.

  MADR decision-makers, consulted and informed. S07.

- [ ] **Never rewrite an accepted RFC; supersede it**

  P16 SCHEMA LINT

  A reversed decision stays in the record, marked superseded, with a link to the RFC that replaced it. It is still relevant that it was the decision.

  Evidence: Rule D1 requires `supersededBy` when status is `superseded`.

  Nygard: keep the old record and mark it superseded. S05.

- [ ] **Link the train and the deprecations, do not copy them**

  P17 SCHEMA LINT

  `target: 2026.10` names the release train (Field Guide 08). `deprecations: [DEP-0007]` names registry entries (Field Guide 09) and each id must appear in the text.

  Evidence: Schema patterns `YYYY.MM` and `DEP-NNNN`; rule X1. The kit's tests resolve RFC-0042's train and DEP id against the other two kits.

  Kit rules.

Status is data. Keep it where the tools can read it.

SECTION 04

Review in the open, decide on a date.

A comment period is a promise to the people who are not in the room: you will hear about this, and you will have time to object.

- [ ] **Comments stay open for five working days**

  P18 LINT

  Count working days, so a proposal opened on Friday does not close on Monday. Five working days sits between React's and Carbon's three calendar days and Rust's ten calendar days, which Rust chose to guarantee five business days.

  Evidence: Rule C1: `commentsClose` is at least five working days after `created`.

  S01, S02, S03; Oxide asks for three to five business days. S04.

- [ ] **Decide after the period closes, and date it**

  P19 LINT

  A decision taken before comments close tells the organisation that commenting is theatre.

  Evidence: Rule D1: `decided` is on or after `commentsClose`.

  Kit rule.

- [ ] **Accepted means scheduled**

  P20 LINT PROCESS

  An accepted RFC names the release train it ships in, and is accepted before that train's decision cut-off: five working days before the code freeze. RFC-0042 was decided on 10 September 2026; the cut-off for train 2026.10 is 29 September.

  Evidence: Rule D2; the cut-off dates come from Field Guide 08's `release-calendar.mjs`.

  Kit rule.

- [ ] **Changes during implementation amend the RFC**

  P21 PROCESS

  The implementation matches the approved proposal. When it cannot, update the RFC in the same pull request and tell the deciders. RFC-0042 was accepted with one change, and the Decision section records it.

  Evidence: The RFC diff appears in the implementation pull request.

  Essay: implementation matches the approved spec exactly.

- [ ] **Keep every RFC in one index**

  P22 PRACTICE

  One folder in the design system repository, one file per RFC, the frontmatter as the index. Rejected, withdrawn and superseded RFCs stay: they answer the question before it is asked again.

  Evidence: `rfcs/` lists every id; a status filter over the frontmatter produces the index page.

  Rust, React and Oxide keep RFCs in a repository. S01, S02, S04.

- [ ] **The linter runs on every RFC pull request**

  P23 LINT

  A non-zero exit blocks the merge. Run it locally before asking for review; the output says what to fix, by rule code.

  Evidence: `node scripts/lint-rfc.mjs rfcs/*.md` in CI and in a pre-commit hook.

  Kit script.

- [ ] **Reviewers check what the linter cannot**

  P24 PRACTICE

  Whether the evidence supports the problem, whether the alternatives were really considered and whether the costs are honest. RFCs that are disingenuous about drawbacks or alternatives tend to be poorly received, and should be.

  Evidence: A reviewer comment on each of the three before the decision.

  S01; Vaidehi Joshi on RFCs losing value when nobody engages. S09.

The linter checks the shape. People check the argument.

APPENDIX A

The template.

`rfc-template.md` from the kit. The linter ignores the HTML comment prompts when it counts words.

rfc-template.md (body; the frontmatter fields are in the table below)

```
# RFC-0000: Short statement of the change
## Problem
<!-- What goes wrong today, for whom, how often. Link the evidence. -->
## Proposal
<!-- The API surface, the code a consumer will write, what is out of scope. -->
## Impact
- Teams affected: <!-- with the usage count you measured -->
- Breaking change: no
- Migration: no
- Costs and drawbacks:
## Alternatives considered
1. Do nothing: <!-- what happens if this is rejected -->
2.
## Decision
Pending.
```

Prompts shortened for print. The untouched template fails the linter on purpose (S2, P1, A1).

Field	Rule	Used by
`id`, `title`	`RFC-NNNN`; 10 to 90 characters; the H1 repeats both (F2)	Index, changesets, registry
`status`, `decided`, `deciders`	Fixed enum; decided date once decided (D1); deciders named	Index, release checklist
`created`, `commentsClose`	At least five working days apart (C1)	Review reminders
`target`	`YYYY.MM`, required when accepted (D2)	Guide 08 calendar
`breaking`, `deprecations`	Agree with Impact (I1); DEP ids appear in the text (X1)	Guide 09 registry

APPENDIX B

One RFC that passes, one that does not.

RFC-0042 is the worked example the other two guides build on. RFC-0043 is the "quick improvement" the essay warns about, written the way such proposals usually arrive.

examples/0042-link-button.md (excerpt)

```
## Problem

`Button` renders an `<a>` when it receives `href`. One component then has
two roles, and the props that only make sense for one of them are accepted by
both. A disabled Button with `href` still navigates (#1712). A loading Button
with `href` shows a spinner on a link that has already navigated (#1760).

A usage scan on 2026-08-31 found `href` on Button in 9 of 41 consuming
repositories, 212 call sites.

## Impact

- Teams affected: 9 of 41 consuming repositories, 212 call sites.
- Breaking change: no. Removing `href` is breaking and ships in 5.0.0 under DEP-0007.
- Migration: yes. A jscodeshift codemod rewrites literal `href` uses.
- Costs and drawbacks: one more component to document.

## Decision

Accepted on 2026-09-10 by the design system core team, with one change:
`LinkButton` has no `disabled` state. Ships in train 2026.10.
```

The Proposal and Alternatives sections are in the kit file. Prior art in the RFC: Atlassian ships a separate LinkButton for navigation [S10].

terminal

```
$ node scripts/lint-rfc.mjs examples/0042-link-button.md
examples/0042-link-button.md: 341/400 words, 23/45 lines, 0 error(s)

$ node scripts/lint-rfc.mjs examples/0043-button-elevated.failing.md
examples/0043-button-elevated.failing.md: 578/400 words, 15/45 lines, 11 error(s)
  line 17  S1  unknown section "Background"; the page has Problem, Proposal, ...
          L1  578 words; one page is at most 400. Cut, or link the detail
  line 27  P1  placeholder "TBD" left in
  line 21  E1  Problem cites no evidence; link the issue, research or usage data
  line 36  A1  Alternatives needs "Do nothing" and at least one other option
  line 31  I1  Impact must state "Breaking change: yes|no" and "Migration: yes|no"
  line 1   C1  comment period closes 2026-09-15; the earliest is 2026-09-21
  line 40  D1  status is accepted but the Decision section starts "Looks good ..."
  line 1   D1  status accepted needs a decided date
  line 1   D2  an accepted RFC names its target release train (YYYY.MM)
  line 1   X1  DEP-0009 is in the frontmatter but not in the text
```

Output from the kit, long lines shortened for print. RFC-0043 also proposes a major change (a new default tone) outside a major window; Field Guide 08's changeset gate catches that one.

.github/workflows/rfcs.yml (steps)

```
on:
  pull_request:
    paths: ["rfcs/**"]
jobs:
  lint:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-node@v4
        with: { node-version: 22 }
      - run: node scripts/lint-rfc.mjs rfcs/*.md
```

Adapt paths to your repository. This workflow was not executed for this guide; the linter it calls was.

KEEP WITH THE RFC

Leave a decision record.

Fill this in when the status changes from `discussion` to a decision. It is the part of the RFC that the next team, and the next agent, will read first.

RFC id / title	**Status / decided on**
Deciders present	**Comment period (opened, closed)**
Objections raised and how each was answered	**Changes required by the decision**
Target release train	**Deprecation ids registered**
Implementation pull request (link)	**Review again if / when**

> **A decision nobody can find did not happen.**
> If the Decision section, the status and the date are not in the RFC file, the next person will reopen the argument from the start.

SOURCES / MAINTENANCE

Keep the guide current.

Sources checked 24 September 2026. Process details (comment periods, states, section names) are quoted from each project's own repository or documentation on that date. The Medium article in S08 blocked automated fetching; its content was verified through search results.

S01 / Rust RFCs, README
When an RFC is needed; ten-day final comment period to guarantee five business days; active, postponed and closed.
https://github.com/rust-lang/rfcs/blob/master/README.md

S02 / React RFCs
Substantial changes only; three-day final comment period; RFCs become active when accepted.
https://github.com/reactjs/rfcs

S03 / Carbon Design System RFCs
A design system running the React RFC process: same gate, three-day final comment period.
https://github.com/carbon-design-system/rfcs

S04 / Oxide, RFD 1: Requests for Discussion
Six states from prediscussion to abandoned; authors, state, discussion and labels as metadata; three to five business days of feedback.
https://rfd.shared.oxide.computer/rfd/0001

S05 / Michael Nygard, Documenting Architecture Decisions
Title, context, decision, status, consequences; one or two pages; superseded decisions are kept (2011).
https://www.cognitect.com/blog/2011/11/15/documenting-architecture-decisions

S06 / Malte Ubl, Design Docs at Google
Context and scope, goals and non-goals, alternatives considered; mini design docs of one to three pages.
https://www.industrialempathy.com/posts/design-docs-at-google/

S07 / MADR, Markdown Architectural Decision Records
Version 4.0.0: considered options, decision outcome, consequences; status, date and decision-makers metadata.
https://adr.github.io/madr/

S08 / Nathan Curtis, Contributions to Design Systems
Small versus large contributions; large ones step through propose, design, code, documentation and release, inspired by Rust and Ember RFCs (EightShapes).
https://medium.com/eightshapes-llc/contributions-to-design-systems-89261a9363d8

S09 / Vaidehi Joshi, Planning for change with RFCs
RFCs as egalitarian, written decision-making; they lose value when nobody reviews them (Increment).
https://increment.com/planning/planning-with-requests-for-comments/

S10 / Atlassian Design System, LinkButton
A separate component for navigation that looks like a button; prior art for the worked example.
https://atlassian.design/components/button/link-button/examples

S11 / Rust RFC template
Summary, motivation, guide-level and reference-level explanation, drawbacks, rationale and alternatives, prior art, unresolved questions, future possibilities.
https://github.com/rust-lang/rfcs/blob/master/0000-template.md

Maintenance: re-measure the page limit if you change the template's headings or your rendering font, and keep the placeholder list in `rfc-policy.json` in step with the template. Recheck the comment periods in S01 to S04 yearly. Update the PDF, HTML, Markdown and JSON together.