[matrix]

April 2021

Demystifying MSCs

Overview

- MSC (Matrix Spec Changes) are how Matrix evolves.
- They're designed to be quick and easy. The TL;DR is:
    - If you want to add a feature to Matrix, just write a free-form proposal.
    - It doesn't even have to follow a template (but we highly recommend it unless you're experienced)
    - Ask for feedback nice and early from the Spec Core Team to check it's something that's wanted (ping @matrix-org/spec-core-team via Github, and/or use #sct-office:matrix.org to ask for attention)
    - PR it into git://github.com/matrix-org/matrix-doc/proposals
    - If it's a work in progress (i.e. not yet ready for review), name it as WIP or make it a draft PR
    - Once you've proven that the proposal works in the wild and you want to get it merged, ask for final review on the PR.
    - Once reviewed, SCT marks it as "FCP merge", giving the community a 5 day warning for final comments (FCP == Final Comment Period), and then the PR is merged.
    - The SCT then turns the MSC into a PR against the spec itself, which is then merged, and is formalised in the spec at the next release.

Writing an MSC

- Think of a change you want to make to Matrix.
- Check to see if it's already in progress: https://matrix.org/docs/spec/proposals
- Sketch out the necessary API changes (HackMD is fab for lightweight collaborative sketches)
- Ideally, check this "proto-MSC" with the SCT to gather initial feedback on whether this is wanted. The best route is to ask for attention in #sct-office:matrix.org. Having got attention, actual discussion should be in #matrix-spec:matrix.org, or MSC-specific room, or Github.
- Assuming you get positive feedback from the SCT, start a PR against the matrix-doc repo. Name your MSC after the PR number Github assigns you (once you open it).
- Read the contribution guide at https://spec.matrix.org/unstable/proposals/. It's not that scary.
- We recommend using the MSC template to help ensure the proposal doesn't have blindspots.
- Go wild and experiment in your favourite client and/or server - see how the changes feel! Before an MSC is merged, you must show a working implementation (to avoid speccing unproven scifi or vaporware). N.B. please check that the SCT is broadly happy before you get too far with your impl.
- However, you **must** namespace API changes with a prefix while you're experimenting to avoid colliding with other experiments (or the final spec, which will likely differ as the MSC evolves).

Prefixes

- You are free to experiment however you like under your own prefix!
- But you can't just go and add unprefixed endpoints/fields, as they would become de-facto spec without actually being reviewed, and without being sufficiently specified for anyone else to be able to implement them, and so clash with other experiments and/or the end result.
- Prefixes should be Java-style. You are welcome to use `org.matrix.msc1234` or your own domain if you prefer e.g. `net.arasphere.threading`. For instance:
  - For an endpoint change: `/_matrix/client/unstable/org.matrix.msc1234/frobnicate`
  - For a field name: `{ "org.matrix.msc1234.frobnicate": true }`
  - For a room version (if you're experimenting with changing how rooms federate): `{ "version": "org.matrix.msc1234.frobnicate" }`
- Prefixes have to have limited lifetime (2 months from spec release) otherwise they effectively become de-facto part of the protocol. Once you are done experimenting & want the feature to become part of the general fabric of Matrix, you must spec & unprefix it.
- This means either discarding any experimental data, or providing a migration path for it (e.g. recreating rooms with the correct m.type).
- If you have a series of dependent MSCs, it typically doesn't matter what order you unprefix in.

Shipping with Prefixes

- One important thing to understand is that **you are allowed to ship implementations which use experimental prefixes.**
- They even don't have to be behind labs flags or labelled beta or equivalent - it is legal for your client to use experimental APIs in order to get features in the hands of users sooner.
- This is critical to allow Matrix to evolve rapidly at a similar rate to its centralised competition.
- **However,** by doing so, you accumulate debt - in that **if you have prefixed data** (as opposed to prefixed APIs) your users will rightfully expect you to implement the complexity to migrate prefixed to unprefixed data once the MSC is specced and merged.
- In other words, you are trading off the ability to ship faster by making life harder in future as you manage the migration needed to unprefix, versus just unprefixing before you ship.
- Sometimes it will be impossible to do a migration (e.g. editing millions of events which have used a prefixed field will be untenable) - meaning you have to unprefix before you ship.
- Sometimes there will be no migration needed at all (if you're just switching from using the prefixed version of an API to an unprefixed version).

Getting it merged

- First of all, do you actually need to get your MSC merged yet? If you're in the middle of implementing a new feature you almost certainly don't need the MSC in the spec - in fact, we won't merge an MSC which doesn't first have a proven implementation.
- However, if you're at the final point of exiting beta for your feature and shipping it to the wider network, you need to make sure everyone else can use it too.
- Hopefully you already got informal review while the PR was being drafted - either spontaneously by people reading MSCs or by asking for it on the PR or in #sct-office:matrix.org.
- Once you want to get it merged, please mark the PR as "ready for review", or comment on the PR, and ping in #sct-office:matrix.org if you like. Do not leave it marked WIP or as a draft PR, as that implies you're not yet happy with it.
- The SCT then reviews it. (In the edge case of the author being unavailable, the SCT may assign a 'shepherd' from the SCT as a proxy). Once the review feedback is resolved, the SCT comments `@mscbot fcp merge` on the PR to propose to merge (or close) the PR. This starts a 5 day "Final Comment Period" (FCP) to give the community a final window to raise concerns.
- Once FCP is over, the SCT hits 'merge' on the PR. Congratulations; you can unprefix & ship! If you haven't unprefixed within 2 months of the relevant spec release, you'll start getting nastygrams from the SCT.

MSC Lifecycle

Fork matrix-doc, copy the template, start writing a MSC. Create a PR (renaming the MSC to match the PR number). Mark the PR as WIP or Draft until you want it reviewed for merge.

PR is now marked as ready for review. SCT review it, and once feedback is incorporated, and any raised concerns are resolved, SCT starts Final Comment Period with intent to merge it.

PR sits in FCP for 5 days. If anyone raises concerns in this time, the FCP is stopped, the concern resolved and then FCP can be restarted.

Having completed FCP, the PR is tagged as “spec-pr-missing” until a Spec PR is written (typically by the SCT) to incorporate the MSC into the actual spec document itself.

Spec PR is reviewed, merged, visible on https://spec.matrix.org/unstable and may be used unprefixed on the wider network. You have 2 months to flush the unstable prefix from your implementations.

Worked example: Spaces

[matrix]

- Drafted in Google Docs, converted to markdown, and then PR'd as MSC1772
- The problem space is large, and should have written a dedicated design doc before then splitting that out into MSCs. Instead, the separate concerns have been factored out from MSC1772 to avoid them blocking each other in review:
  - How to group rooms into sets of rooms called spaces (remained as MSC1772)
  - How to manage power levels based on space membership (MSC2962)
  - How to restrict access to rooms based on space membership (MSC3083)
- The core MSC1772 defines four new identifiers, with the following prefixes:

Proposed final identifier	**Purpose**	**Development identifier**
type	property in m.room.create	org.matrix.msc1772.type
m.space	value of type in m.room.create	org.matrix.msc1772.space
m.space.child	event type	org.matrix.msc1772.space.child
m.space.parent	event type	org.matrix.msc1772.space.parent

Worked example: Spaces

- During alpha and beta, testing space trees have been created using prefixed events and room types.
- We could ship with prefixes, but we'd have to provide a migration path. As room types are immutable, this would be disruptive as we'd have to recreate the rooms. (We'll have to create private rooms anyway when MSC3083 lands to bump the room version, but the disruption is avoidable for public rooms. That said, we do need to ensure room upgrades are not disruptive...)
- So instead, we decided to get MSC1772 FCP'd before shipping, so space rooms are created with the final unprefixed type (`"type": "m.space"`) and won't have to be migrated.
- (We experimented with splitting out the type definition into a separate MSC3137 to unprefix types so as to ship sooner, but it's not really meaningful to specify an "m.space" identifier without saying what it does, so instead we FCP'd MSC1772 instead).
- Finally, MSC3083 needs a new room version (to define the "restricted" join_rule) - so before it can ship, we will need to define and release that room version and upgrade rooms whose membership is defined by spaces to use it.

MSC Tips

- This presentation isn't a substitute for the full guide at the top of https://spec.matrix.org/unstable/proposals/ - if you are writing MSCs you **must** read this!
- Get review as early and broadly as possible; nothing worse than spending ages on a proposal only to discover it's a duplicate or is otherwise ill-fated.
- Keep MSCs succinct and as high signal-to-noise as possible (while remembering to give sufficient context for the motivation). Do not waffle.
- Remember that anyone is free to comment on PRs and wider comments may or may not be well balanced. Talk to the SCT if a commenter is being unconstructive; we will moderate.
- HackMD is awesome for collaboratively sketching out MSCs before they hit Github.
- Try to keep each MSC relatively small; if it's getting too big, refactor! This makes it easier to review, and lets you ship incrementally rather than risk having one contentious section blocking the whole MSC from merging. If you refactor, **please explain why**.
- Layer features in Matrix. With clear levels of abstraction, we should be able to evolve and swap out bits of the stack without impacting others.

Who actually is the Spec Core Team?

[matrix]

- The SCT is defined at the bottom of https://matrix.org/foundation/
- As of April 2021, the membership and nominal scope is:
  - Matthew Hodgson (Lead, Guardian)
  - Erik Johnston (Servers)
  - Richard van der Hoff (Servers, Cryptography)
  - David Baker (Clients, IS API, Push API, VoIP, Media)
  - Hubert Chathi (Cryptography, General)
  - Andrew Morgan (Servers, AS API, Spec Process)
  - Travis Ralston (Spec PRs, Bots and Bridges & AS API, Media)
  - Alexey Rusakov (kitsune) (Clients on behalf of Community)