2021-08-30

- Go no go? Discussion notes
- Overall stats
    - 872 re

2021-08-17

- What kind of FCP do we require?
- Stabilizations
    - Disjoint capture in closures
    - Reserving syntax
    - What procedure we want to follow, do we feel need an FCP
- Docs, meta-set
    - Edition Guide
    - Reference
- Crater issues
    - Want to block on ICEs
    - Migration doesn't allow code to run with wrong semantics or affects a large number of crates (>5% of tested crates)
- Migration issues are generally not blockers
    - We can fix them in subsequent releases
- Suggestions that don't work in some edge cases, not critical
- Go-no-go for 1.56 is next Monday
    - Before that we need: to be in FCP for stabilizing lang features
    - Before that we need: pending PRs for known ICEs
- Target next monday for crater -- get as much done as we can -- to avoid noise

2021-06-28

- Blockers for the public testing period:
- Blog post calling out for public testing
    - Just something short
-

2021-06-21

- Prefix--
    - Maybe just land it and have petrochenkov review asynchronously
- Force-warn
    - Needs to be stabilized
- Disjoint captures
    - One blocking bug w/ PR
    - Need to tie the use of the feature of the edition
- Need to land the PR that updates the lint groups
- Crater run, what should it do?
    - Still need this

2021-06-14

- Reserved prefixes
    - Code seemed pretty good, niko has to sync up with matklad / petrochenkov and then r+
    - Take edition from span
        - To make a test
- Migration for prelude
    - Need to get this to something that can be merged
- Lints and force-warn
    - There is a per edition “backwards incompatibility” group
    - So long as they have the edition mentioned in the metadata
    - Tracked by 85894
- Disjoint capture closures
    - Not tied to the edition flag
- Crater run, what should it do?
    - Modify cargo fix so that if has a `--if` flag to only do something in 1 edition
    - Modify cargo test so that
        - It runs `cargo fix --all --edition`
        - If no crates upgraded, skip test
        - If there are errors, abort
        - Force edition to 2021 and then run tests as normal
    - And save this at some point

2021-06-01

- Force warn:
    - Ryan and I will work through that
- RFC 2229
    - Haven’t auto-migrated a bunch of crates and see the edge cases
    - Trying to measure size, but that doesn’t block our immediate milestone
    - R+ 85724 and then I think we are good to go
- Migration for prelude
    - Makes some false positives
    - Has some unhandled
    - Also gives bogus suggestions
        - `<chalk_ir::CanonicalVarKinds as CanonicalVarKinds<I><_>>::from_iter
- Reserved prefixes
    - Edition dependent lexer or not
        - String escaping is controlled by the lexer and that actually hardcodes quite a number of decisions
        - Examples of things that we can’t do
            - c”\xXX” -- not require UTF-8 here
                - Otherwise we would require c#”\xXX”
    - Cases we’ve come up with
        - `match”foo”` -- error

        - Token string function in procedural macros
            - Has a defined behavior which seems correct
            - Should have a function that takes a span
    - Core problems
        - Lexer today lexes strings in ways that are not desirable for some future extensions we have in mind
            - For example, `f”ffff{“”}` cannot accommodate nested quotes (which some folks want, others don’t)
            - C strings cannot embed non-utf8 escape sequences
            - Byte strings today refuses non-ASCII characters, we can’t model this in the lexer if we wanted it elsewhere
        - Lexer currently has no knowledge of editions
            - → True but fixable, it will mean that the procedural macro lexing functions may want to add variants that accept spans to control what edition is used
        - There exists code that uses keywords as “prefixes” `match”foo”`
            - → Not a problem, insert a space
        - Macro rules arm ordering: if we grow the set of legal expressions, this can be affected
            - Lex error prevents us from getting here
            - Jointness means we always produce same set of tokens
            - But if can change the meaning of `$t:expr`
            - Jointness cannot be expressed in macro-rules, cannot take multiple tokens and preserve jointness
                - What happen with `+=` today?
    - Non problem
        - If we used jointness, someone could create a procedural macro that makes it a breaking change for us to add a prefix
            - Not true-- in procedural macros you have to upgrade to a new syn to see this behavior
    - Next step:
        - Niko to write this up and post it on the issue
- Where are we at?
    - Not quite as far as we wanted to be
    - But in some ways better off:
        - New prelude has data from a “pre-lint crater run”
    - Worried about the disjoint thing, less and less time to discover problems
    - What do we need to do to do a “giant crater run”
    - Documentation? Ground work is basically laid, extending this off seems fairly easy
        - What does it take to get the migration guide get published?
        - https://doc.rust-lang.org/nightly/edition-guide/rust-2021/index.html
-

2021-05-20

- Rylev and Niko chat a bit about items
    - Edition guide;
        - Merge https://github.com/rust-lang/edition-guide/pull/232
            - **nikomatsakis**
        - **rylev** needs to read over the “overall copy” and clean it up
            - Ping steveklabnik after
        - Each group needs to review their corresponding page
            - **rylev** to coordinate
    - Force-warn
        - Plan is in the gist
        - **rylev** to make MCP out of gist
- Other stuff
    - https://github.com/rust-lang/rust/issues/84513 has some cleanup work

2021-05-10

- Prelude changes
    - Needs migration work
    - https://github.com/rust-lang/rust/issues/84594#issuecomment-830689847
- Reserving prefix syntax
    - Open question about token trees
        - What span to use for the string
        - Currently defined as callee site
            - This may not behave as desired since the callee of the proc macro may be some
        - Maybe definition site of proc macro makes more sense
            - Then the behavior doesn’t change if you upgrade client, in case that the string was internal to the proc macro
            - Otoh if the string came from client, as in quote!, not right
        - Or maybe just rust 2021
            - We don’t expect actual breakage from this change, and in the future we could bump it if necessary
        - Right behavior is we still need to have a span
    - We should implement
- Finish migration guide work

2021-05-03

- Lints
    - PR is r+’d, just getting all tests to pass
- Prelude changes
    - Migration plan is in place
    - Migration implementation required
    - RFC is awaiting a checkbox
- Or-patterns

    - Maybe the lints aren't setup right for migration, investigate
- RFC 2229
    - https://github.com/rust-lang/rust/pull/84730 marks end of migration work, needs review
- IntoIter
    - The main impl work is done, but there is cleanup work to be done
    - https://github.com/rust-lang/rust/issues/84513
- Reserved syntax
    - RFC needs to be merged
    - Impl work needs to be done
- Blog post status
    - In review
- Edition guide
    - We need to restructure to have highlight the changes
    - For each change, we would want to link to a "motivational" description of the surrounding feature
        - But the RFCs are often not a great resource
        - Either outdated or just..not great
        - In some cases, non-existent
    - Blog post parts are a great starting point!
        - We can use those!
        - Add just add edge cases
- Multi-span issue
    - Eric mentioned reverting some PRs
    - But the revert is complicated, there are multiple PRs involved and an issue with a lot of text, Mara feels a bit uncertain
    - https://github.com/rust-lang/rust/issues/53934
    - Is this an issue with lints that actually exist, or just theoretical lints?
    - Summary, machine applicable:
        - If MachineApplicable: All MachineApplicable must be applied
        - Otherwise: may be exclusive and may not be what the user intended
    - Next step:
        - Write-up the summary #53934 ← Niko
            - https://gist.github.com/nikomatsakis/d067b2ef324525eae510532d0dca325b
        - https://github.com/rust-lang/rustfix/pull/156

2021-04-26

- Lints
    - Issue has been resolved, PR is passing CI, needs to land, has been approved
- Prelude changes
    - Dirkjan and NIko agreed on the general plan
    - **Need to open an issue with the plan**
- Or changes

    - Need to land the PR that updates the name: https://github.com/rust-lang/rust/pull/83386 <- `@bors r+`
- Disjoint capture in closure
    - Work proceeds
    - Working on the last bit of migration
- IntoIterator for [T; N]
    - Merged
- Rustc-args-with-const-generics
    - Not going to make any Edition-related changes
- Reserving prefix syntax
    - Resolved the RFC status
    - **Need to get cracking on the implementation; there was some kind of PR…**
    - Who should we ping about this/
- Rustfix multi-span issue
    - Joshua was almost volunteering for it
    - If we have different lints for the automation, we can just use a single span
    - There may be a fix from before, which was reverted
- Prep the edition migration guide
    - Niko started this but found it was a bit non-trivial
    - Ryan was planning to do it but has a list of work
    - Niko may chop things down poorly so that they have the right structure but wrong content
- Ignore allow(some_migration_lint) in cargo fix mode
    - Officially on Ryan's plate, which is overfull
    - We need an MCP
- Blog post:
    - Incomplete draft: https://hackmd.io/@m-ou-se/BkoVpBVD_
        - (Rendered in rust blog style: here)
    -

2021-04-19

- RFC
- Lints
    - Some kind of problem is blocking rylev from landing the PR
    - As of right now there are no lints outside that PR under consideration
    - There was talk of using the edition to move some FCW to hard errors
        - (lints that have widespread impact, even if breakage would technically be allowed)
    - Niko is ok with us slipping on this :) we have enough on our plates
        - Really what this is about is a post edition work item to make a process around FCWs; this has been a desiderata for a long time, but we lacked bandwidth and organizational capacity
- Prelude

        - No progress, still need a migration plan, ideally implemented by May 1
        - Main problem
            - Methods from new traits that may now be ambiguous
            - Suggest a rewrite to the `foo::bar(self,
- Or patterns
    - `pat_param` name was proposed and is waiting on FCP
- Disjoint capture in closures
    - Great progress on the last few code complete items
    - Niko to review PRs
    - Rustc has been using it for some time
- IntoIter
    - Pending PR in FCP now that fixes the problem by tweaking method dispatch
    - Migration lint is basically done
        - Would be good to have write-up for rustc migration guide
- replace rustc_args_required_const with const generics
    - This probably won't happen for the edition but that's ok
- reserving # related syntax
    - Where we are at:
        - Josh wants to permit select prefixes (e.g., `f`)
        - Mara would like to give the power to proc macros, at the cost of having to select our lexing rules now
    - Alternative proposal:
        - Keep it a lex error
        - Request RFCs to open up specific prefixes
            - Either as non-lex-errors
            - Or with full semantics
- Blog post summary / outline
    - Repeat general edition plan, not breaking changes
    - Here are the candidate items for the 2021 Edition
    - Next steps in the process
        - All implementation and migration lints aim to be completed by May 1
        - Write-up the migration details in the rustc migration guide
        - Around June 15 there should be a guide available and a call for testing
    - Things that did not get in and why
        - We realized we didn't need an edition
    - Want to help?
        - Write-up specific issues where we need help
    -

2021-04-12

- Lints
    - We need to work on rustfix to ensure that it will ignore the lint settings
    - Have to look at what happens with multiple groups but probably some kind of override setting will fix it
- Prelude changes
    - Still stuck on migration plan; djc, but haven't gotten any updates

        - Have a crater run that shows all the failures but what kind of migration lints should we be giving
        - Action item: follow up with djc and/or try to recruit someone new
- Or patterns
    - Just the settle bikeshed
- Clojure capture
    - Continued progress, basic migrations working
- IntoIter
    - We can actually do this by modifying `.` notation
- Replace rustc_args_required_const with const generics
    - Little bit of a bikeshed but either way the edition can happen
- Reserving prefixes
    - Should macros error out on the reserved space at lex time?
    - What about prefixed literals? (e.g., `f”foo”`)
        - Quite a few DSL already react to syntax like this
        - In Rust 2021, you would have to write `f “foo”`
            - Some proc macros actually look at the whitespace
        - Summary of challenges:
            - If we tokenize,
            -
    - If we DO tokenize
        - We have to migrate with a space to ensure code doesn’t break
            - People have to update their procedural macros
        - For literals (notably strings):
            - Right now there are times when knowing what sort of string it is must be tokenized
            - For example, byte strings etc
    - Deadline: Have one week to settle it
        - Niko to do a decent write-up and summarize the considerations
-

2021-03-29

- Lints
    - Stalled out, too much going on
    - PR https://github.com/rust-lang/rust/pull/83213
    - That needs to merge but the rest of the ongoing work doesn’t block edition shipping, by and large
- Prelude changes
    - Consensus about the changes but no real migration plan
    - Were waiting for crater but it’s been broken
    - https://github.com/rust-lang/rust/pull/82217 was merged
    - Who’s working on it? Djc, haven’t heard any updates lately
- Cargo resolver change

    - Done and done, has warnings, etc
- Or patterns
    - Issue is open, volunteer working on it, PR is getting close
        - Migrations
    - Uncertainty about the actual names we should use
        - Pat-atom seems ok
    - We expect there to be a challenge around issuing suggestions, you have to add this metadata, and issue it later, but mara hit some sort of panic because it was allow-by-default
        - Complained that not all lints are processed
- Disjoint capture in closures
    - Decided to pursue `let _ = &x`
    - Work is in progress now
    - On track to get all the impl work done in 1 week or two
    - Plan to do some size measurements after that to judge impact
- Unscoped rustdoc lint names
    - Might be just a lint
    - No good reason to error in this edition necessarily
    - Don't have very clear guidelines here
- IntoIter from [T; N]
    - Niko will try to do this today
- New range types
    - Won't get resolved in the next few days
    - No real change, still two sides and no clear consensus
    - "Absolutely should have different types"
    - Need someone to do a write-up or something
- replace rustc_args_required_const with const generics
    - Hard work is done, final decision around compatibility is pending, but whichever way is ok
- k#foo
    - More discussion, RFC is kinda done, seems like we're just stuck on some minor-ish points
- Reserve `f"..."` syntax?
    - Reserve "syntactic space"
- Goal: Write a blog post this week
    - Want to announce the final set of candidates
    - We could start writing it but we don't know the final results
- Edition migration guide
- Uncertainties -- will we include this?
    - k# and f"" foo
        - Needs an RFC -- @scottmcm?
    - Rustdoc lint names
        - Final decision: are we just going to warn for now? (Seems ok)
    - Range types

        - Two camps, need someone to do a write-up to try and then have a meeting of the minds of lang/libs to see where people fall, Niko can advise
    - IntoIter from [T; N]
        - Niko needs to investigate, likely not edition tied since impls are not edition tied

2021-03-22

- Prelude changes
    - Crater run, still waiting on that
- Cargo resolver change
    - “Completely done”
    - When you upgrade and you don’t have a resolver explicitly set it will warn you about the diffs
- Or patterns
    - Opened https://github.com/rust-lang/rust/issues/83318 and somebody volunteered
    - Have to get pat2015 and pat2021 worked out
    - Maybe propose an alternate name --
        - Let’s discuss in the Zulip topic
- Disjoint capture in closure
    - Migration impl in progress
    - `let _ = &x`
    - `|| expr` => `|| { let _ = &x; expr }` -- probably will do as one big diff
        - But might be nice not to
    - Crater run to assess impact in various ways
- Rustdoc attributes
    - Plan of record:
        - Error in new edition when `#[foo]` is used
        - Support `#[rustdoc::foo]`
        - Warn in older editions
    - Needs a migration
- New range types
    - Discussion doesn’t seem to be converging, deadline fast approaching
    - Everybody seems to have different opinions
        - Major downside of current types
            - Range is not Copy
            - RangeInclusive has extra fields
- Replace rustc with const generics
    - Current iteration
        - Const generics accepted syntactically in the normal place
        - This will break everyone’s code
    - Option 1, Option 2

        - Option 1: support hack forever
        - Option 2: stop supporting in rust 2021
    - Option 2 looks nice for Rust, but it means a divergence between how intel manual describes it
    - Report that Amanieu wrote
        - One very confusing thing is that sometimes write
            - `foo(0, 1+1, 2+2)`
    - Libs opinion:
        - New edition only accepts proper rust syntax (2nd one)
        - But produce a warning with a machine applicable fix from the 1st one
    - Next step: discuss in lang meeting
- `k#foo` --
    - Kind of stalled, niko to ping scott
    - Talk about in lang team meeting
- Edition guide
    - There is material in current guide that may not exist in the reference
        - For example, explanations of the module resolution changes
    - Maybe open an issue with the content and suggesting moving it somewhere else
    -

2021-03-15

- Things we want to get volunteers for
    - Rustfix multi-span thing -- esteban
    - Migration for or patterns -- niko
    -
- Unify std/core panic
    - Blocked on the rustfix bug
    - Oli says probably not hard to fix but he doesn't have time to work on it
    - Esteban replied with an emoji
    - We need to find somebody
- Lints
    - Ryan plans to open PR soon, seems like most of the machinery we need exists
    - Future incompatibility lints -- do we want to use the edition for this? We need to discuss
- Prelude changes
    - Landed a PR that creates the edition preludes
    - Going to measure breakage if the new editions were available in both preludes
        - We know there will be some breakage
        - Want to know in which ways things break
        - Goal is to get concrete examples of what breaks in practice
        - Possible uses:
            - To figure out how best to write edition migration guide
            - Maybe to give heads up to folks

        - To figure out whether automated migration (rustfix) is needed
- Cargo Resolver Change
    - Haven't heard anything new here, basically all done except for doc questions
- Or patterns
    - Breakage and migration:
        - Plan was to require manual migration because breakage is relatively narrow
        - But we do know of some breakage
    - Question for Niko: verify that pat2018 and pat2021 exist
        - Naming point: pat2015 -- name after the first edition
    - Would be fairly simple to do more precise warnings, Niko thinks, but hasn't gotten around to writing up a plan
    - "Tens of root regression"
- Disjoint captures in closures
    - drop(&a); -- existing workaround
    - #[captures(a, ..)] -- proposed RFC syntax
    - `|| x + 2` => `|| { drop(&x); x + 2 }`
    - Discussion thread on zulip -- hackmd with draft RFC
- Deprecate unscoped rustdoc lint names
    - Is there an intersection between editions?
    - Or is there confusion about Rust versions?
- IntoIter from [T; N]
    - Start a thread and ping nikomatsakis and matthew jasper
- new range types
    - Stephen was going to update the draft RFC with new feedback
- replace rustc_args_required_const with const generics
    - Amanieu just posted an update: `stdarch` const generics: I plan on updating the submodule this week, which should hopefully net us some compilation time improvements. However we're still not sure whether to make this an edition breaking change. This depends on the libs team deciding whether we want to allow the legacy const generics syntax in the new edition (`_mm_shuffle_ps(a, b, CONST)` vs `_mm_shuffle_ps::<CONST>(a, b)`).
    - Options
        - (A) Gentle: accept it everywhere, issue a warning
        - (B)
- `k#foo` or some similar mechanism
    - Gotta get this up like yesterday
    - Some syntax questions
    - Gotta get the PR up
- Never type fallback
    - The plan is
        - Make a hybrid that will allow us to change infallible to `!`
        - Land this on all editions
        - Start to warn for "scary bits", maybe these become hard errors in 2024
    - TL;DR we are not doing this for the edition
- Prep the edition migration guide
- Lints
    - Future-incompatibility warnings-- used to signify code that is not currently an error but which will become an error in the future

    - Machinery in the compiler to produce warning
        - "In some future version"
            - Used for bugfixes
        - "In edition X"
            - Official idioms -- array into-iter
            - Unofficial idioms -- panic lint
                - Warns "hey we will be changing this in a future edition'
    - Lang team write-up
        - https://hackmd.io/i1Ob4XS6TwuUv-rOVEoM4A
    - In some cases we may want to leverage the edition
        - Warning
        - Deny by default in all editions
            - Can be skipped under extenuating circumstances:
                - But what are they? If warning is too crude, maybe?
                - Code too hard to maintain?
        - Optional: Hard error in new edition
            - We stop here if:
                - Large breakage in practice by making a hard error
                - Maintenance burden to keep old code working is not too high
                - Risk n practice from existing instances of this code is not too bad
        - Error in all editions
    - Next steps:
        - Which FCW (if any) do we wish to make a hard error in the new edition only
        - Name questions to definitively resolve:
            - What is an FCW?
            - What do we call the other things that are currently FCW but maybe not in the future?
            - Do we keep using the term FCW for anything?
        - Write down the policy (incorporate into draft RFC?)
            - Naming changes

2021-03-08

- Unify std/core panic
    - Blocked on the rustfix bug
    - Ehuss
- Lints
    - Seems like only 2 are clearly edition specific
        - Go ahead and submit / nominate a PR for those
        - Want to turn them into hard-errors
    - Future compatibility warning
        - Maybe we want to sometimes make these into edition errors because of widespread breakage
    - Revising the RFC for lang team policy
- Prelude PR
    - Edition part was discussed last week
    - Doing a crater run, only thing missing is a concrete migration plan

    - What kind of warnings do we want to give?
- Cargo resolver change
    - Migration is done
    - Question: Edition migration guide
- Or patterns
    - Niko didn't figure out what is going on
- RFC 2229
    - Progress
- Deprecate unscoped rustdoc lint names
    - Main body of the work is done but needs migration plan
- IntoIter for [T; N]
    - Niko has to think about whether this can be added as an edition
- Range types
    - Still haven't settled the question of how to manage the name
    - Is there a way
- k#foo
    - Scottmcm wrote a draft RFC
- Never type fallback
    - Mark is reviewing
-

2021-03-01

- Unify std/core panic
    - Applying automated changes is blocked on rustfix not supporting multispans (#53934)
        - JSON doesn't have a way of communicating AND vs OR
        - Could workaround this by avoiding multispans
    - Ehuss is working on some other related things, but not that as far as we know, maybe ask them about it
- Lints
    - Had a meeting
    - Made some concrete decisions, also worked towards a more general policy
    - Can start advancing the PRs for those decisions
    -
- Prelude
    - Most of the things in the RFC are not edition related
        - Mostly not traits, won't break anything
    - The edition changes are small, libs team can probably make a decision there fairly quickly
        - Adding TryFrom, TryInto, FromIterator into the prelude
    - Migration plan is needed
        - Can do a crater run to try and assess impact
        - Have a stability-related warning, can maybe leverage but for stability

        - Can leverage the fact that it's a known set of method names if we have ot
- Cargo resolver change
    - Resolver 2.0 change will become stable in 1.51
    - Change is to make 2.0 the default
    - The warning plan from the issue seems good
    - Does ehuss plan to do this or should we find someone else? Rylev to follow up
- Or-patterns
    - How smart can we make the migration?
    - Niko to ping mark-i-m and see how much time he has to put into this
- Disjoint capture in closure
    - Wanted to use a `std::marker::captures!(a)` macro
        - Expands to `drop(&a)`
    - Niko to do a write-up of this and open an issue
- Deprecate unscoped rustdoc lint names
    - Still activity, mara to check
- IntoIter from [T; N]
    - Not clear how this can work over an edition
    - Niko to weigh in on the options
- new range types
    - Want to change what `..` expands to (something that's copy, but not iterator)
    - Hard to do something nice -- don't want to just add something to stdlib, what to call it?
    - Do we want to have a "redirectable" type alias? Or do we want to just come up with another name?
    - Another option: just implement `Copy` for `Range` type, but that is something we've always avoided
    - Still need migration lint because `..` expands to something else
    - Another option: add the "must clone" warning
        - Libs team is not a fan
- remove macro_export/pub macro_rules
    - Have enumerated some use cases that are not possible or easy in the new system
    - Recursive macros are difficult, particularly when the macro doesn't know the path to itself
    - Trying to decide how big of an issue these things are
    - Seems like we need to offer a way to use the "textual scope" macros, how do we do that?
        - Re-opt in
        - Default if not marked with pub?
- replace rustc_args_required_const with const generics
    - Who even owns this? Oli?
    - Niko to ping oli
- deprecate/remote static mut
    - Removing from the list; if we do this, we'll start with a warning

- No warning for an associated constant always hidden by an enum variant #76347
    - Why is this an edition?
    - Niko to check with Scott
- k#foo mechanism
    - Point of this is to make features available earlier than the next edition, if not in a super ergonomic way
    - Needs an RFC
- Reserve yeet as a keyword
    - Not happening
    - Niko moved this to the end of the list, not Mara
    - Jane: blame Niko
- Never type fallback
    - Changing this is something Niko really wants to do
    - But can be also change Infallible
    - Does it make sense to leverage edition to minimize breakage?
    - Niko to prepare a report on impact or find someone to do so

Follow ups

- Ryan talked with ehuss about cargo resolver work:
    - ***Eric Huss****: Everything on that list is done and already on nightly except for the part that compares the resolve graph to warn about differences. That will be a bit more challenging, but I'll probably give it a try at some point. I don't have an estimate when that will be done, though.*
    - ***Eric Huss****: When stabilizing 2021, there are some steps that need to be done on both rustc and cargo. I documented (most of) the cargo steps here.*

Topics for future discussion

- Ehuss had a question related to documentation (specifically the edition guide):
    - ***Eric Huss****: I'd also like to know a little more about what's going on with documentation. I asked earlier in the Zulip stream, but didn't get a response. No rush, but there seems to be some strong assertions in the RFC about documentation, but it is not clear how that is going to be done.*
    - *From the RFC: "We maintain an Edition Migration Guide that offers guidance on how to migrate to the next edition."*
    - *Questions: Who is writing that? Is it an extension of the existing guide? Will the existing guide change in structure?*