Integrations Processes

Overview

Once we hit the Hardlock milestone in a given release stream we begin enforcing an integration process for any late commits.

Common Resources:

- Integration Submit Tool Guide
- Temporary Allowlister Process
- Permanent Allowlister Process
- Integrations channels:
    - #fn-integration-requests-ext
    - #ue-integration-requests-ext
- For any questions/issues around SubmitTool:
    - #dev-build-health-tools-ext

On this page:

Release Team Training Deck: Integrations (Part 2 in the Deck)

Sign in to your Google Account

You must sign in to access this content

Sign in

Release Team Session Recording: Integrations

Link to session recording here.

Allow-listers

Temporary Allow-listers

- In cases where we have big beats or specific targeted features for releases, temporary allow-listers can be added to review and submit their own changes only up until Pencils Down.
- It is expected that there is sound reasoning for allow-listers to be added, and that production and lead engineers are monitoring their teams.
- Temporary allow-listers are usually assigned for a business need (like an LTM release) to help keep work moving and relieve possible bottlenecks.
- Temporary allow-listers are only able to submit for their given areas, while permanent can submit for anything in the branch.
- Process: Temporary Allow List Process

Permanent Allow-listers

- Permanent allow-listers stay on the list, release after release, and have to be vetted by a Technical Director to be added to the list.
- They can submit their own work *and* the work of others.
- Permanent allow-listers can be added and have the ability to check-in to all hardlocked FN branches.
- This is normally for component owners, engineering leads, or other approved allow-listers for a given team.
- This requires the fn-techdirector group to sign off on additions to the permanent allow-list.
- Process: Permanent Allow Listed Access

Integration Expectations: Pre-Live

Hardlock Expectations

- All integrations have a Jira associated with:
  - FixVersion set to the proper release/branch they are targeting
  - Priority of the ticket is a Priority 0/1
- Pre-flights are preferred to validate the changes are safe late in the release, as well as embedded QA sign off.
- Reviews are required from allow-listers, and can be submitted by allow-lister or Release Management once approved.

Pencils Down Additional Scrutiny

- Pre-flights and embedded QA testing are necessary beside very specific edge cases.
- Severity/Impact has to be high (Ship Blocking Issues Only).
- Expected: Cvars around late changes that give us hotfix levers to disable if needed.
- Only Release Managers can submit changes into the branch past Pencils Down.

Integration Expectations: Post-Live

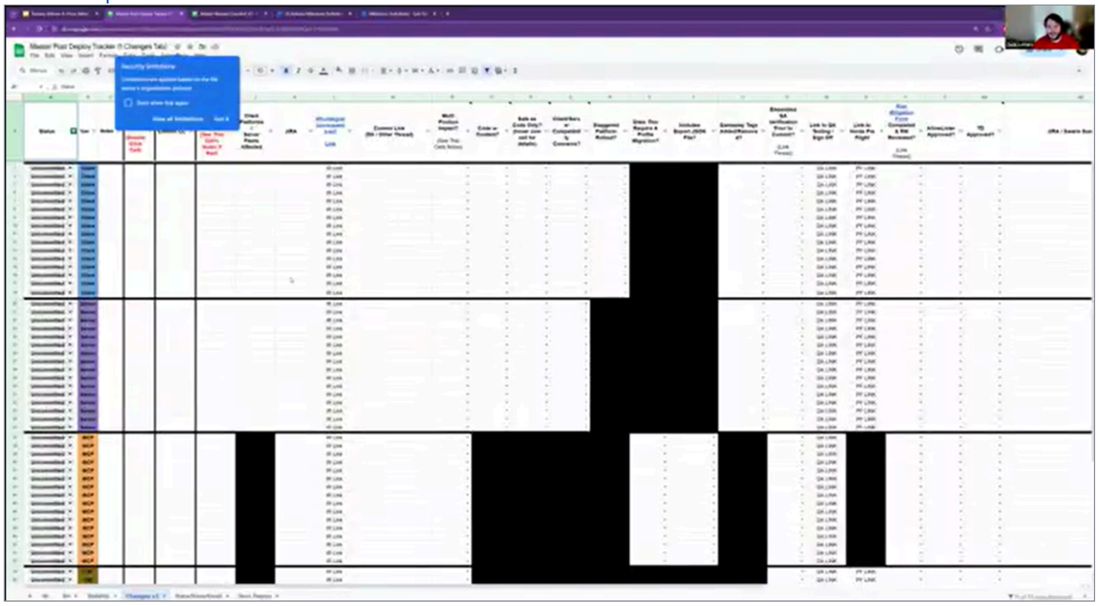
Once we are Live we need to be exceptionally cautious about any integrations in an already live stream (or soon to go live). Close coordination between operations RMs around the planned EP slots and timings is essential.

- Still following the same integrations process of ensuring review, pre-flights, testing, etc.. but not submitting until completely aligned.
- This is to ensure that we leave room for absolutely critical items to be committed & turned around quickly without additional risk from other changes.

> ⓘ I.E. if we pull in changes as they are requested they would exist in the branch and any future builds. The more changes we have in a build, the higher likelihood of one of the changes jeopardizing the entire build.

For any changes we are tracking or for any kind of EP's, we track them in a Post Deploy Tracker: Post Deploy Tracker - Template

› Click to expand a tracker screenshot...



- Though this is for Post Deploy tracking, we normally have some preSH's or early EP plans to track prior to launch.

We track changes in the Changes v2 tab with required fields so that we understand how to properly handle each change. Key information includes Jira, Swarm, Integration Thread, SH/Coordination link.

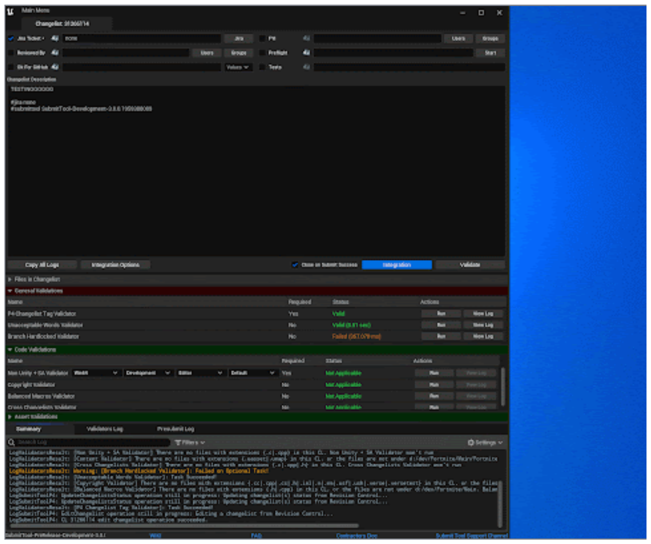
For EP facing integrations, we need to ask some critical questions:

- Which Clients and Server Fleets does this change affect?
- Is there any Multi-Product impact to consider?
- Is this a Code or Content change? And is it safe for a Code-Only Build?
- Any Client/Server Compatibility Concerns?
- Can this change be in a staggered rollout? (Client Only Question)

How Integrations are Submitted

The main flow for submitting integrations is through the SubmitTool:

- Submitter will utilize the SubmitTool to provide a swarm shelf in the desired stream. The shelf will be presented with required fields to submit alongside the swarm:
  - Which product does the change impact (sub-questions may apply based on selection).
  - Associated Jiras, impacts, risks, etc...
- Both FN and UE use this system:
  - If UE is selected the request will post to #ue-integration-requests-ext
  - If FORT is selected the request will post to #fn-integration-requests-ext
- Integrations SubmitTool Guide
- Not common, but if there are issues with SubmitTool a manual request can be submitted through the ERM service desk: Manual Integration Form



Main Menu

Changelist: 31266714

Changelist Description

```
TESTINGGGGGG

#jira none
#submitted SubmitTool-Development-3.8.6 7059088088
```

General Validations

Name	Required	Status	Actions
P4 Changelist Tag Validator	Yes	Valid	Run / View Log
Unacceptable Words Validator	No	Valid (0.01 sec)	Run / View Log
Branch Hardlocked Validator	No	Failed (567.079ms)	Run / View Log

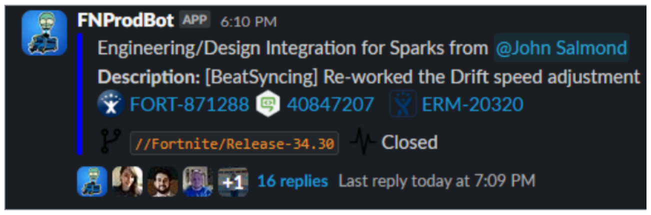
Code Validations

Name	Required	Status	Actions
Non Unity + SA Validator	Yes	Not Applicable	Run / View Log
Copyright Validator	No	Not Applicable	Run / View Log
Balanced Macros Validator	No	Not Applicable	Run / View Log
Cross Changelists Validator	No	Not Applicable	Run / View Log

```
LogValidatorsResult: [Non Unity + SA Validator] There are no files with extensions (.c|.cpp) in this CL. Non Unity + SA Validator won't run
LogValidatorsResult: [Content Validator] There are no files with extensions (.uasset|.umap) in this CL, or the files are not under d:/dev/Fortnite/Main/Fortnite
LogValidatorsResult: [Cross Changelists Validator] There are no files with extensions (.c|.cpp|.h) in this CL. Cross Changelists Validator won't run
LogValidatorsResult: Warning: [Branch Hardlocked Validator]: Failed on Optional Task!
LogValidatorsResult: [Unacceptable Words Validator]: Task Succeeded!
LogValidatorsResult: [Copyright Validator] There are no files with extensions (.cs|.cpp|.csj|.h|.ini|.m|.mm|.usf|.ush|.verse|.versetest) in this CL, or the files
LogValidatorsResult: [Balanced Macros Validator] There are no files with extensions (.h|.cpp) in this CL, or the files are not under d:/dev/Fortnite/Main. Balan
LogSubmitToolP4: UpdateChangelistsStatus operation still in progress: Updating changelist(s) status from Revision Control...
LogSubmitToolP4: UpdateChangelistsStatus operation still in progress: Updating changelist(s) status from Revision Control...
LogValidatorsResult: [P4 Changelist Tag Validator]: Task Succeeded!
LogSubmitToolP4: EditChangelist operation still in progress: Editing a changelist from Revision Control...
LogSubmitToolP4: CL 31266714 edit changelist operation succeeded.
```

Integration Process

- Once submitted a slack thread will be generated to coordinate the integration. An ERM ticket will be generated with each request.
- RM's responsibility is to monitor for open integrations, loop in and coordinate with allow-listers, and once aligned, to submit the integration.
- To submit:
  - Open the swarm link identified, and ensure that you have allow-lister sign off on the swarm or in the thread.
  - Select Approve and Commit from the dropdown.
  - If you hit any errors trying to submit, respond back to the thread and coordinate any additional changes needed to the description prior to committing.
- The ERM ticket attached to the integration thread is what drives the status of the integration thread.
  - Once a change is submitted, reply back to the thread with the submitted changelist, and the ERM ticket should be closed with the changelist information.



FNProdBot APP 6:10 PM

Engineering/Design Integration for Sparks from @John Salmond

Description: [BeatSyncing] Re-worked the Drift speed adjustment

FORT-871288 40847207 ERM-20320

//Fortnite/Release-34.30 Closed

+1 16 replies Last reply today at 7:09 PM

Integration Process: Reviews

In general we rely heavily on the allow-listers and leads to properly review each change, but from an RM side there are few key areas to consider.

Do the files make sense for the given change?

- Quick check to verify that no changes were included by accident.

Does the change have localization impact?

- All player facing text requires localization to a number of languages. Localization team needs to know as soon as possible for any string changes after hardlock due to turn around.
- If localization is not possible, modifications to the integration may be requested.

Will it cause a large spike in patch size?

- Our goal is to keep patch size to a minimum by working closely with Tech performance and the patching team to catch any issues.
- With new Content Patch and determinism, keep a close eye on anything that will cause patch sizes to increase dramatically.

Is there a generic/master shader file being touched? If yes then ask about invalidation expectation.

- Small changes in 'Base' files can significantly affect a build system-wide. In many cases, a late change may invalidate some work by QA and Performance teams processes, which will result in redoing the work.