

Current Process From Code To Release

Mobile Platform

19 November, 2025

Table of Contents

1 Notes on Automation vs. Semi-Automation Reasoning.....5

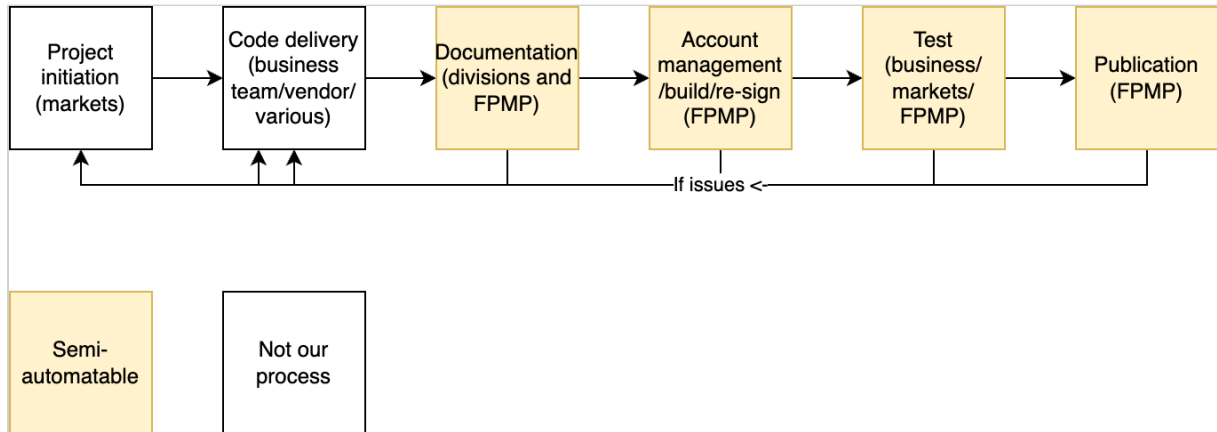
1.1 Semi-automation:.....5

1.2 Automation:.....6

Notes:

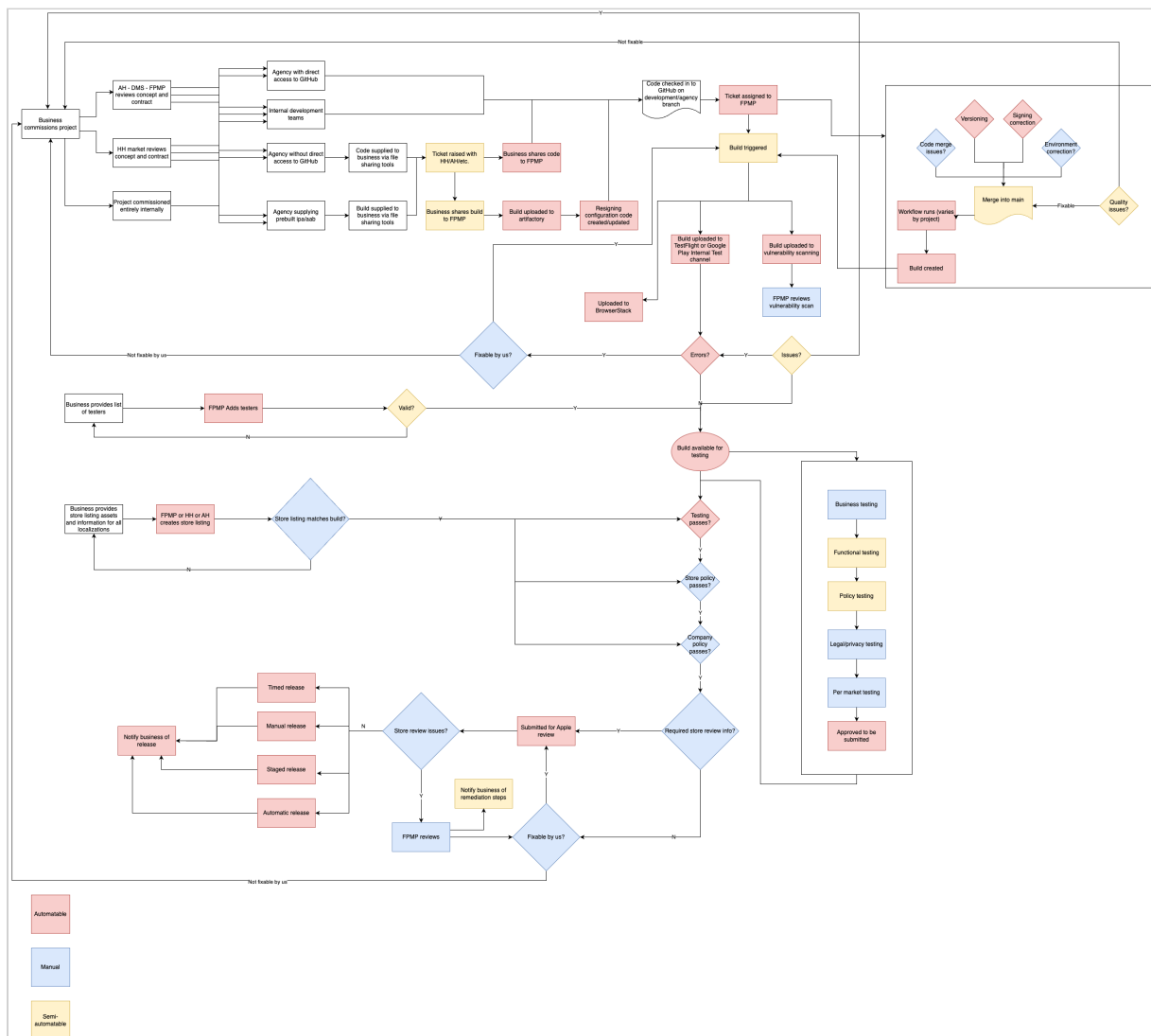
[Notes on Automation vs. Semi-Automation Reasoning \(see page 5\)](#)

High level diagram:



Describing the current process from code being received by our company to release:

Mobile Platform – Current Process From Code To Release



1 Notes on Automation vs. Semi-Automation Reasoning

1.1 Semi-automation:

Ticket raised with AH (internal and public apps)/HH (public apps) → mature system that should not be touched, possible to automatically link masp.merck.com ticket and respective Service Desks

Business shares build to FPMP → as above, if builds shared in tickets, possible to automatically upload to artifactory, however some manual data will need to be entered (e.g. which ticket is it for, which app - app names vary across systems, versioning, what is being shared - sometimes vendors supply naked ipa or aab and sometimes zip file and sometimes just an apk and we have to chase them for actual builds and sometimes is a link to vendor's file sharing system sometimes is just a text file saying to go to vendor's file sharing system, etc.)

Build triggered → for anything which requires code to be merged or versioning/signing/etc. corrections then manual intervention will be needed, but once done the trigger itself can probably be handled the same as the Resigning trigger is.

Code merging → some conflicts are automatically fixable by GitHub, others aren't, which branch/etc. or even whether to merge code into a release/main branch depends on each solution's branching and development strategy. Many pitfalls possible.

Vulnerability scan / issues → some elements could be automatically flagged, but others will always require manual review. Possible to detect *if* an app has a vulnerability scan started, however many possible issues with automation due to how DataTheorem's MobileSecure works (e.g. possible to have same bundle ID and same version and same build listed twice with different issues open due to prod vs. pre-prod handling).

Tester population → depending on platform and supplied data. Historically business has not understood that they need to have an Apple/Google ID, Google will block tester addition if not a Google ID, or even the difference between ISID and Apple/Google ID, etc. Could upload list for Apple at least and hope, but always requires some manual review.

Testing → apps are meant to be used by humans, therefore they will always need to be tested by humans. Automation of testing possible but it can only test for known scenarios, and can't perform exploratory or full acceptance testing. Similar with policies - we can check for presence/absence of certain elements (e.g. if a link to privacy policy is present, etc.), but depends on policies and requirements which naturally change over time (and Apple/Google aren't always notifying us ahead of time).

Notify business of remediation steps on Apple/Google review issues → some elements can be automated, depends on the error and involvement of business. Note Apple and Google both use manual *and* automated testing in their reviews, and their documentation emphasizes the need for manual testing of apps.

1.2 Automation:

Business shares code to FPMP → handled by GitHub == mostly automated by GitHub's processes depends on branches, we'd just need to figure out the triggers

Build uploaded to artifactory from MASP (resign workflow) → already automated by Dan

Resigning of build → already automated by Dan (NB manual intervention will always be needed at some point)

Build uploaded to TestFlight/GP Internal Test/Vuln scan/BrowserStack → already automated on Bitrise workflow (NB Google blocks automation of upload if there are policy issues in the metadata so exception handling is manual but as long as we have the error message then we can act on it)

Build available for testing → automatable using Apple's APIs, GP's system is designed to require manual review

Testing passes → possible to have notification pathway in MASP (e.g. user changes status on MASP to passed and it notifies us, but carries with it some problems with user management)

FPMP creates store listing / metadata population → possible to automate using APIs from VSMU to Apple/Google. (with some limitations)

Submit for Apple Review → possible to automate from MASP to Apple/Google using APIs (as long as metadata is correct and all store policies are conformed to, requires manual review in some cases). Google's APIs not so friendly

Release method → possible to automate from MASP to Apple/Google using APIs (with some limitations).

Release succeeded → possible to automate from Google/Apple APIs or just from scrapes (would requires checks against last vs. current)