[Sample][PRD] Centralized Force Update Mechanism for Android and iOS Apps

Initiative owner	@Ferry Irawan
UI/UX Researcher	
UI/UX Designer	
Data team	
Commercial Stakeholder	
Epic	SPEX-160: Deliver Centralized Force Update Mechanism for Android and iOS Apps TO DO
Updated Details	• Updated Date [date] • Updated By @Ferry Irawan • Update Summary ◦ Initial Document
Status	Error loading the extension!
Approval or Reviewed	☐ [Product Lead] ☐ [Engineering Manager or Tech Lead] ☐ [Quality Assurance] ☐ [UI/UX]
Related Docs	• [RFC] • [UX Research result]

Target Population

Target Description

- Qualitative description of what target segments your initiative is designed for
- Exp:
  - Active companies that have employees > 100
  - Companies that create PPN & Witholding tax in their transactions

Target Description

- Estimated % of active companies that this initiative will have impact on

User Value

User Problem Definition

- Qualitative description of user problem definition - use specific persona for who is having this problem
- Exp:
  - Accountants have problem to slice & dice financial reports based on sales from specific location and expenses by departments
  - Bookkeeper need to give information on transactions accurately to help Accountants do slice and dice of reports

Current State

Our mobile application ecosystem faces several critical challenges related to version management and user adoption:

Fragmented Update Enforcement

- Each of our N mobile applications implements force update mechanisms independently with inconsistent logic and thresholds
- No standardized approach exists across Android and iOS platforms, leading to varied user experiences
- Teams spend redundant effort building and maintaining similar force update features

Version Fragmentation Risk

- Users running outdated app versions (6+ months old) pose significant risks including security vulnerabilities, degraded performance, and incompatibility with backend services
- Without centralized visibility, we cannot accurately track version adoption rates across our portfolio
- Manual intervention is required to enforce updates, creating operational overhead and delayed response times

Root Causes

- Lack of centralized infrastructure for version tracking and update policy enforcement
- No automated mechanism to trigger update prompts based on version age
- Absence of standardized remote configuration system across all apps
- Manual processes for monitoring and enforcing update compliance

User Problem Frequency

- How often the target users experiences the problem the feature is designed to solve

Soft nudge: Apps 4+ months old show update reminders

Strong recommendation: Apps 5+ months old show persistent update prompts

Force update: Apps 6+ months old require mandatory update

User Problem Severity

- How important is the problem to target users
- Level of severity
  - High
  - Middle

- Low
- Define also why you think this problem is on that specific level
- Exp:
  - High - because accountants don't know weather a financial reports is already final on a certain period and there are no more pending transaction or account transactions that is still being calculated on the background

Business Value

Business Impact

Value added by feature to the business:

- Acquisition - acquiring new users
- Retention - retaining existing users
- Monetization - monetizing users

This should list lagging indicator and Product Metrics

- Support burden increases as outdated versions generate more crashes and compatibility issues
- Feature rollout velocity is limited by the long tail of users on legacy versions
- Security patches and critical fixes take longer to reach 100% of the user base
- Inconsistent update experiences damage brand trust and user satisfaction

Strategic Impact

Importance of feature to the product value proposition

- Defensive - Because our competitor need it or if not provide customer will churn
- Offensive - capture new market & add value prop compare to competitors

- Defensive - Because we can reduce issue reports from old versions and ensure highest security within the latest available apps.

Proposed solution

- Define the proposed solution as clear as possible. "Show don't tell"- Put in Mockup or user flow when needed
- Define impacted modules and other products / team
- Technical documentation link (if available)

Description & Flow

[Force Update Flow Here]

Mockup & Design

[Design of the Force Update Dialogs]

Technical Documentation

Solution Overview

Implement a centralized, automated force update mechanism that leverages remote configuration and CD pipeline integration to enforce version policies consistently across all N mobile applications on both Android and iOS platforms.

Core Components

1. **App Versioning Catalog (Automatic)**

- Integrated into Ohara CD pipeline to automatically capture and store version metadata
- Records: `app_id`, `platform`, `version_number`, `build_number`, `release_date`, `deployment_status`
- Serves as single source of truth for all app versions across the portfolio
- Provides APIs for version lookup and age calculation

2. **Standardized Force Update SDK/Module**

- Reusable library integrated into all mobile apps (Android + iOS)
- Handles three update tiers:
  - **Soft Nudge** (4+ months): Dismissible reminder with "Update Now" and "Later" options
  - **Strong Recommendation** (5+ months): Persistent modal requiring user acknowledgment, reappears after 3 app launches
  - **Force Update** (6+ months): Blocking screen preventing app usage until update is installed
- Consistent UI/UX patterns across all apps with customizable branding
- Telemetry hooks for monitoring user responses and update completion rates

3. **Remote Configuration Service**

- Centralized configuration management for force update rules
- Parameters: `soft_nudge_threshold`, `strong_recommendation_threshold`, `force_update_threshold`, `grace_period_hours`
- Supports per-app override capabilities for edge cases
- Real-time configuration updates without app redeployment

4. **Auto Validator Service**

- Scheduled job (runs daily) that:
  - Queries App Versioning Catalog for all active app versions
  - Calculates version age from release_date
  - Automatically updates remote config flags based on 4-5-6 month thresholds
  - Generates alerts for versions approaching force update deadline
  - Produces compliance dashboard showing adoption rates by app and platform

5. **Monitoring & Analytics Dashboard**

- Real-time visibility into version distribution across all apps
- Tracks update completion rates for each tier (soft/strong/force)
- Alerts for apps with <80% compliance on current or previous version
- Historical trends for version adoption velocity

Success Criteria

- What are the things that will be considered as a **leading** indicator that this initiatives are done and successful
  - Feature adoption target (% of active users)
  - Feature retention target (retention % by natural frequency)
  - Feature satisfaction target (by CSAT or interview)
- Also put tracking progress here

Version Compliance

- **Target**: 100% of active users on apps <6 months old by [target date]

Automation Coverage

- **Target**: 100% of N mobile apps integrated with centralized force update mechanism

Update Response Rates

- **Soft Nudge**: ≥40% of users update within 7 days of prompt
- **Strong Recommendation**: ≥70% of users update within 14 days
- **Force Update**: 100% compliance (blocking prevents usage)
- **Measurement**: Update completion rates tracked via telemetry

Constraints & Limitation

- Define what capability, feature or design that will not be covered on current initiatives
- Put duration of time box if we want to have this initiatives time-boxed

Supporting Documents

GTM Checklist Template (doc)

- Please duplicate the document template and create your own GTM Checklist
- Reference for GTM criteria (doc)

NEPD Form Template (doc)

- Please duplicate the document template and create your own NEPD Form
- Reference for NEPD guideline (doc)

User Stories

User Stories

App Versioning Catalog

Deliverable 1

Deliverable 1	Enable Mobile team to have an app versioning across all apps for every release pipelines.

Description	Allow each apps to register: • App Name • Platform • Bundle Id • App Version Name • App Version Number • Release Timestamp • Release Changes → Instruct Robin to execute Me-Doc • Release Notes (can be edited after release)
Milestone / Success Metrics	1. Version metadata is captured on every production deployment in Ohara 2. Catalog stores app_id, platform, version, build number, and release timestamp 3. API endpoint available to query version information by app_id and platform 4. Historical version data is retained for audit purposes
Dependencies	-

#	User Story/Scenario	Importance	Mock up	Acceptance Criteria
Officeless Studio				
1.	**As a** Mobile DevOps Engineer **I want** the CD pipeline to automatically log app version and release date to the catalog **So that** we maintain an accurate, real-time inventory of all deployed versions without manual intervention	MUST HAVE		1. Version metadata is captured on every production deployment in Ohara 2. Catalog stores App Name, Platform, Bundle Id, App Version Name, App Version Number, Release Timestamp 3. API endpoint available to query version information by app_id and platform 4. Historical version data is retained for

				audit purposes
2.	**As a** Mobile Platform Developer **I want** an API that calculates the age of any app version **So that** the Auto Validator can determine which update tier applies	MUST HAVE		1. API accepts app_id, platform, version as input 2. Returns version age in days based on release_date 3. Handles timezone normalization correctly 4. Response time <100ms at p95
3.	**As a** Mobile Product Engineer **I want** to be able to see my mobile apps version catalog inside Ohara Dashboard **So that** i can use the data to know which version is released when and contains what	MUST HAVE		1. Add 1 menu Version Catalog in Ohara Sidebar 2. Add 1 screen showing the version tables 3. Add filter toggle to enable user to select app version catalog based on the app name 4. Add input field to be able to update Release Changes and Release Notes to be editted after release

QA Analysis

QA to provide feedback based on PRD, RFC and design analysis

- Ensure the design index provided will not impact to application performance
- Check correlation between product and stock adjustment to show correct calculation after implementation

Changelogs

Date	Name	Notes