Comprehensive Design Document for Profile Lifecycle Management

# RAID (Risks, Assumptions, Issues, and Dependencies)

## Risks

1. Current ME profile lifecycle management has third-party relationships and group roles that are not part of CM lifecycle management.  
- \*\*Mitigation Plan\*\*: Business lending will take over groups as the target source of record. Specific third parties like mortgage introducers are not included in CM's target scope.  
2. AdvantEdge has a batch interface with ME where inactive statuses are sent to ME, but CM does not get the status directly from AdvantEdge.  
- \*\*Mitigation Plan\*\*: In the long term, CM will master these interfaces directly, getting updates from AdvantEdge.

## Assumptions

1. CM will manage inactivation events and downstream systems need to consume these updates accordingly.  
2. Customer Master only contains customer data while archival and deletion processes are managed by respective business units.

## Issues

1. Manual inactivation/reactivation processes are not currently prioritized, and this could delay individual case handling.

## Dependencies

1. The lifecycle management depends on integration with Arrangements API to verify account activity status for relationships.

# Solution Options

## Out of Scope

1. Changes that involve new downstream systems beyond existing NAB infrastructure.  
2. Lifecycle management for non-CM mastered profiles, except for basic compliance alignment.

## In Scope

The solution includes profile lifecycle management for CM-mastered customer profiles across multiple brands, including Redstar, White Label, and NAB brands. Manual deletion, merge capabilities, compliance with group retention policies, and automated lifecycle processing are part of the scope.

# Solution Architecture Diagram

The solution architecture will include components responsible for handling the customer lifecycle, leveraging Java services to identify profile updates, manage relationships, and coordinate inactivation or purging of profiles based on conditions.

# CAST Architectural Requirements

CAST requirements will be needed for any new application introduced to ensure it aligns with NAB’s security, data flow, and operational volumetrics standards.

# Solution Details

The design details for the profile lifecycle management consist of the following modules:

## 1. Initial Job Script Handling for Inactive Candidates

- \*\*Functionality\*\*: Identify parent and child profile entities that haven't had updates in the last two years.  
- \*\*Approach\*\*: Use batch job scripts that execute periodically to evaluate profile activity based on metadata and transaction timestamps.  
- \*\*Technical Flow\*\*:  
 - Step 1: Load profiles with no updates from the last two years.  
 - Step 2: Verify the status of any linked child entities.  
 - Step 3: Mark profiles that meet the criteria as 'inactive\_candidate'.

## 2. Consumer-Based Relationship Handling

- \*\*Functionality\*\*: Identify relationships involving customer profiles to determine if they should remain active.  
- \*\*Process Flow\*\*:  
 - Step 1: Retrieve all relationships for a given customer profile.  
 - Step 2: Validate each relationship with external services and account systems.  
 - Step 3: Break relationships if they do not have any active accounts linked.  
- \*\*Interaction\*\*:  
 - Consumers call the Arrangements API to check for any active accounts associated with the relationships.

## 3. Arrangements API Integration

- \*\*Purpose\*\*: Ensure that customer profiles that have no active relationships are marked for inactivation or deletion.  
- \*\*API Flow\*\*:  
 - Step 1: Trigger the Arrangements API from the consumer application.  
 - Step 2: API fetches all arrangements (accounts/services) tied to the given customer.  
 - Step 3: Return the status of each arrangement (active/inactive).  
 - Step 4: Make inactivation decisions based on returned statuses.

## Java-Based Sequence Diagram

The sequence diagram will demonstrate the interaction between the profile management module, consumer services, and Arrangements API. It highlights the following interactions:  
- \*\*Initial Batch Job\*\*: Triggers identification of inactive profiles.  
- \*\*Consumer Service\*\*: Calls API to verify relationships and active accounts.  
- \*\*Arrangements API\*\*: Returns statuses that determine profile handling (inactivation, retention, or deletion).

Diagram Placeholder: High-Level Solution Architecture Diagram will be included here.

Diagram Placeholder: Sequence Diagram will be included here for Java-based service flow.

# Implementation Guidelines

The implementation of the Profile Lifecycle Management solution includes the following guidelines:  
- \*\*Batch Processing\*\*: Job scripts will be scheduled periodically (e.g., every 30 days) to assess profile activity status.  
- \*\*API Integration\*\*: All external API calls should be secured using OAuth2 for ensuring data integrity.  
- \*\*Error Handling\*\*: Fail-safe mechanisms should be in place for batch jobs, including retry logic and error logging.

# Risks and Considerations

1. Dependency on third-party APIs such as the Arrangements API may cause delays if the external service is unavailable.  
2. Manual inactivation processes are not prioritized, which could delay individual customer requests.

# Implementation: Change Scope and Sizing

This section will provide details of changes required in existing components, including sub-component breakdowns, implementation cost estimates, and resource allocation.