Channel Info Store

Technical Document v1.1
Integration Services – Strategic Solution Approach

**Service Discovery**
- Systems and Applications Analysis
- Integration Endpoints Analysis
- Feasibility Study
  - Build the framework components

**Design**
- Solution Architecture and Design
- Customise the below framework components as per the integration services
  - API components
  - Common Components
    - Error & Retry Handler
    - Audit Handler
    - Libraries and Routines
  - Kafka Solution and design

**Development**
- To Build the below integration services
  - In scope of Project requirement
  - Integration Services
    - Error & Retry Handler
    - Audit Handler
    - Libraries & Routines
  - Build of Kafka Services
  - Build of APIC Services

**Package**
- Micro Service Application
- Container Routines
- Business Orchestration Services

**Deploy**
- Deploy - Corporate
- Deploy - Finance
- Deploy - Retail

**Check In**
- Git Version Control
  - Feature Branch
  - Release Branch
  - Dev Branch
  - Master Branch

**Trigger Process (Sequential)**
- Analyse Code
- Compile Code
- Sanity Check
- Package

**Binary Version Manager Repo**
- Dev Repo
- Stage Repo
- Prod Repo

**Env**
- Dev Env
- UAT Env
- Prod Env
Problem Statement in a typical bank

**Problem**
- Delay in response times during peak business hours
- Frequent timeouts
- Poor Customer experience
- Situation worsens during pay day weeks on a monthly basis
- Real time access of historical data is a challenge
- Making real time data available for multiple reporting channels

**Impact**
- Loss of customers due to poor experience
- Roadblock to digital enablement goals
- Creates hinderance to onboard time-to-market channel business features
- Delays to apply new regulatory changes imposed by regulatory body
Problem Statement: Digital Journey in Banks are Challenging Why?

- Banking Core Apps
  - Increased Infra resources to meet new demands from Channel

- Retail Mobile Banking
- Corp Banking
- ATMs
- 3rd Party
- Payment Wallet
- Limit Management

Core & Channel Interface Middleware

Banking Core Apps – Infra resources Increased to meet new demands from Channel

Scaling legacy core apps remains complicated & expensive

Tightly dependent with core business units and operations

Ends up spending huge money to add more licenses & increase Infra resources
In general, below are the frequently used entities in a banking system. For a new implementation, it is suggested to take the read related channel interface services and then move to more complex business functions for ex: transaction related.

<table>
<thead>
<tr>
<th>01</th>
<th>02</th>
<th>03</th>
<th>04</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer</td>
<td>Accounts</td>
<td>Cards</td>
<td>360 Degree View</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>05</th>
<th>06</th>
<th>07</th>
<th>08</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transactions statements – historical And daily</td>
<td>Online Money Transfers</td>
<td>Batch Payments</td>
<td>Lending</td>
</tr>
</tbody>
</table>
Real-time Channel Integrator: Proposed Approach Reference

Any changes to Customer, Accounts and Cards Info [Create/Update/Delete]

1. Core-Bank

Changed entity Info (Accounts, Customer, Cards Info)

Synch any changes as events to table

2. CDC based Adapter

Core Apps DB

Datastores

Datastores

API/ESB

3. For Real-Time & Incremental - Message Transformation + Orchestration

CACHE

Channel Info Store

4. Request/Response

Channel Services

Monitoring & Management Support
Integration Platform Modernization – New Components and Services

01. EntryMsgFlow

02. ExitMsgFlow

03. ErrorHandlingSubFlow

04. Database Calls Logic

05. Logging Framework

06. Application Framework

07. Logs Backup and Restore

08. CI-CD automation

09. Containerized platform

10. Real time monitoring - ELK

11. Backend calls using Kafka streaming

12. • Dynamic scaling
   • Dynamic Routing

13. Improved Agility
   Faster resolution time to identify and fix any failures/faults

14. • Ease support of Managed Services
   • Swiftly create integration services
Key Benefits

01. Re-designed service flow with lean framework
02. Improved consistency & average response time
03. ~2 times improvement in volume of data processed
04. Scalable Environment
05. Speed, Flexibility, & Agility
06. Offload reads from core system
07. DevOps Pipeline automation
08. Real Time monitoring Support - ELK
09. Future Proof Framework and resusable components
Thank You