BIAN Reference Implementations

BIAN
Banking Industry
Architecture Network

Known BIAN Implementations – 2020[1]

Service Management[2]

- PNC
- Santander
- SPD Bank (&IBM)
- CIBC
- ABSA Bank (& TCS)

APIs

- PNC
- IBM
- SPD Bank (& IBM)[3]
- ABSA Bank (& TCS)[3]
- Santander
- RedHat

BOM

- SPD Bank (& IBM)
- ABSA Bank (&TCS)

Devops CI/CD

- PNC
- RedHat
- SwaggerHub (&GitHub) – API Compliance Tools

Open Banking

- IBM SD Mapping

Certification[4]

- Pilot
  - Temenos
  - DXC
  - Interested
    - Intellect Design
    - Union Bank of Philippines

[1]Implementations that we are aware of through webinars, working groups and discussions; likely to be more in reality.
[2]Application Landscape Heatmaps, Application Portfolio management, Business Architecture, Business Capabilities.
[3]Further implementations are possibly underway, that can't be disclosed for confidentiality or other reasons.
[4]Certification (DRAFT) Working Paper & Questionnaire (Spreadsheet) have been produced, being reviewed.

https://bian.org/participate/bian-webinars/

PNC

ABSA (&TCS)

IBM

SPD Bank (& IBM)

RedHat

Santander

CIBC

Solution Certification

API Governance, Compliance & Feedback

HIGHLIGHTS

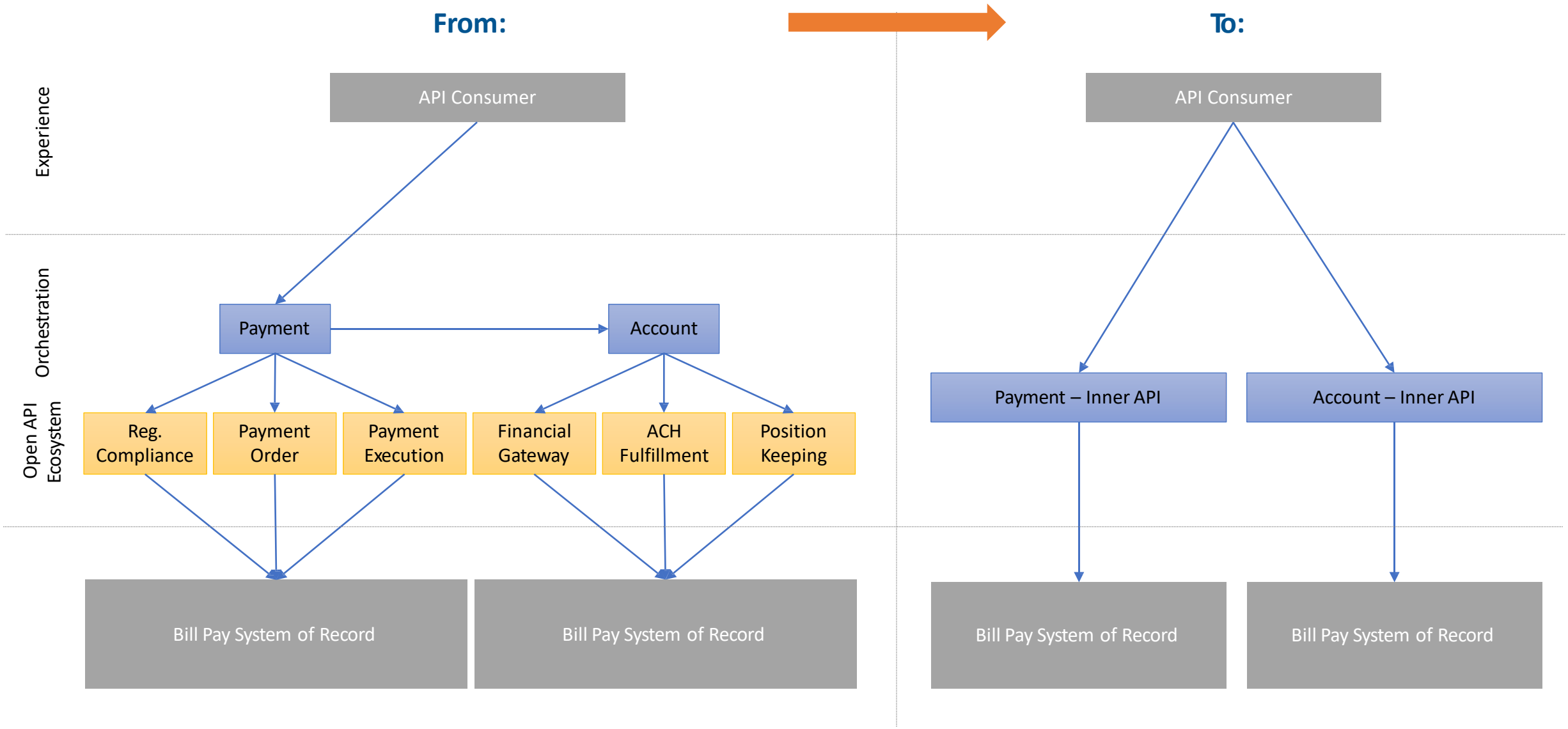
PNC

July 28, 2020

ADOPTION OF BIAN APIS

BIAN Inspired APIs

Moving from fine-grained endpoints to logical endpoints representing business functionality



Adoption of BIAN APIs

PNC's new architecture strategy includes 'Inner APIs' which perform a single function and align to BIAN standards

Experience

Fine-grained representation of a user interaction. Specific user function, only concerns itself with performing that function as best as possible

Outer API

Tightly coupled to the UI, provides the data necessary for display purposes by the UI. It is the Outer APIs job to filter out unnecessary data and to **orchestrate** invocations across many Inner.

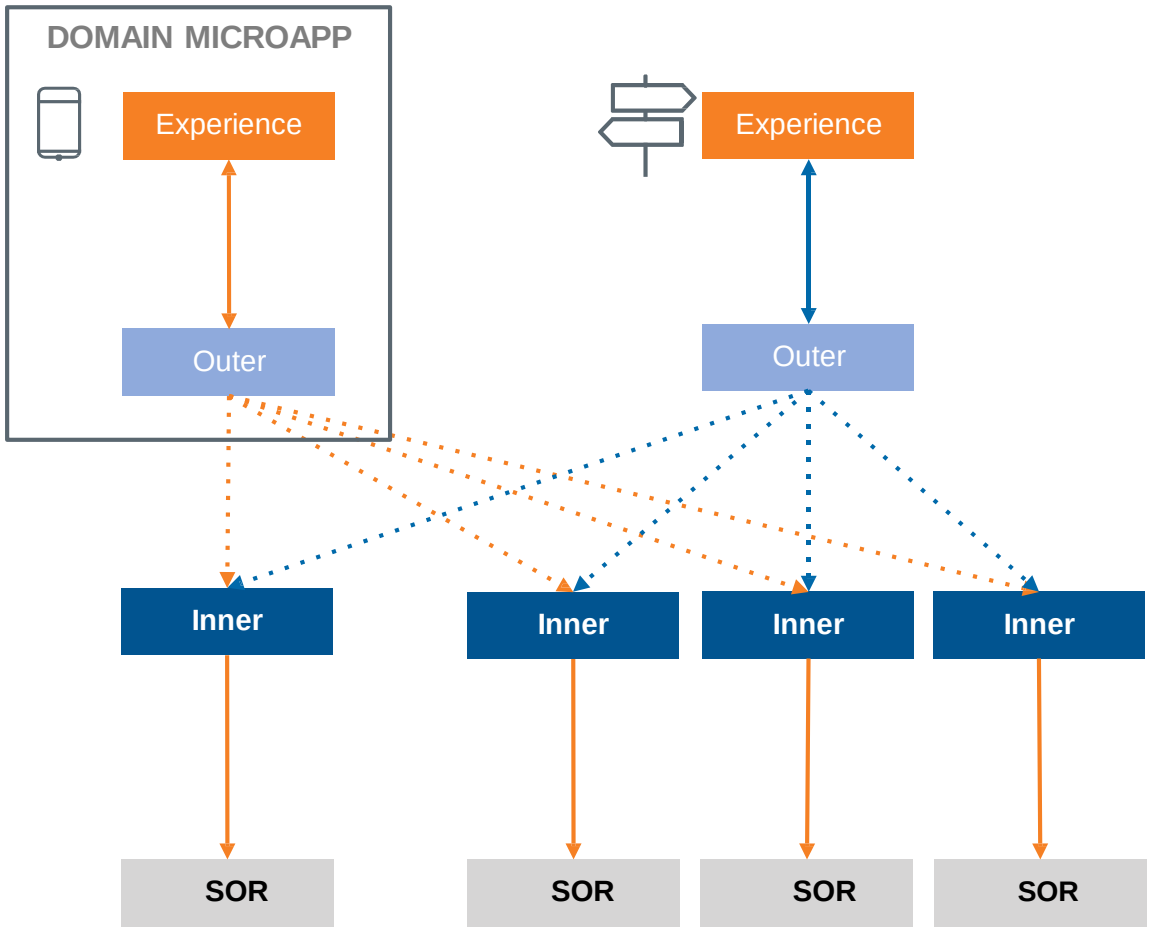
Inner API *(BIAN Inspired)*

Exposes all functionality and data contained within a SOR to the enterprise. The Inner API's evolution is tied to the evolution of the SOR and should only **perform one function,** allowing for abstraction from the SOR

Inner APIs are inspired from the BIAN API Model and leverage the BIAN Object Model (BOM) to model their inputs/outputs

- To avoid vendor lock-in and provide a BIAN inspired Inner API, we must build our own Inner APIs
- This principle is important to abstract out Vendor Systems of Record
- An Inner API will need to be built that calls the Vendor's proprietary API

New PNC Architecture



DEVELOPING WITH BIAN

Developing with BIAN

PNC has developed a number of tools to integrate the BIAN framework and data modeling standards across the development lifecycle

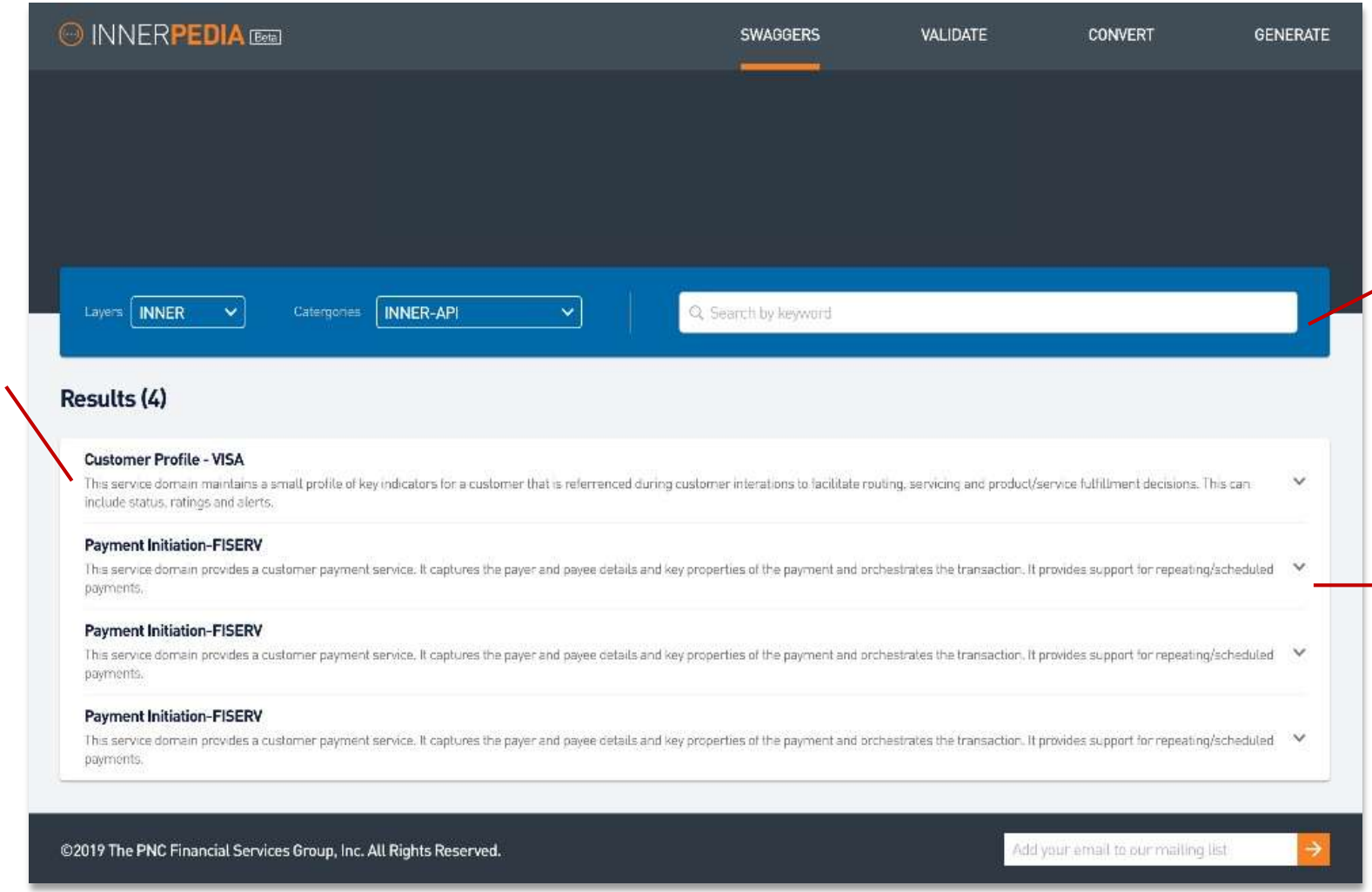
	API Discovery	API Toolkit	Generator & Validator	Pipeline Enforcement	Compliance Dashboard
	Innerpedia	**Endpoint Catalog & API model**	**Swagger Generator & Validator**	**GitOps Pipeline**	**Compliance Dashboard**
					
Description	Catalog of endpoints, inspired by the BIAN endpoint console, for individuals to discover in-development and completed endpoints	Mechanism for users to select BIAN-inspired PNC endpoints and build their payload from already modeled entities and properties	Tools that generates and auto-populates swaggers with data from the API Toolkit while ensuring adherence to BIAN and PNC standards	Pipeline outfitted with various enforcement capabilities to ensure standards are enforced	Compliance dashboard aggregates microservices' adherence to leading practices
Benefits	▪ Promotes re-use ▪ Provides all Inner APIs in one unified location ▪ Allows users to understand how to leverage and call specific services	▪ Improves time-to-market by accelerating development cycles ▪ Enables seamless integration and data sharing across systems	▪ Accelerates modeling and development time in-line with contract-based development ▪ Enables users to better self-serve and check for adherence to standards	▪ Automated pipeline checks ensure applications meet our set standards ▪ Enforcement mechanism to ensure applications not modeled correctly do not reach production	▪ Increased visibility into application compliance ▪ Alerts simplify management of 'drifted' applications

Innerpedia

Innerpedia is PNC's central repository for inner APIs, allowing users to search and understand the capabilities of each of API while promoting re-use and adherence to BIAN naming conventions

Inner API Naming

Applications follow the standard naming convention of: **'BIAN Service Domain - SOR**'



Searching

Type keywords here to search for any existing APIs that perform the service you are looking for

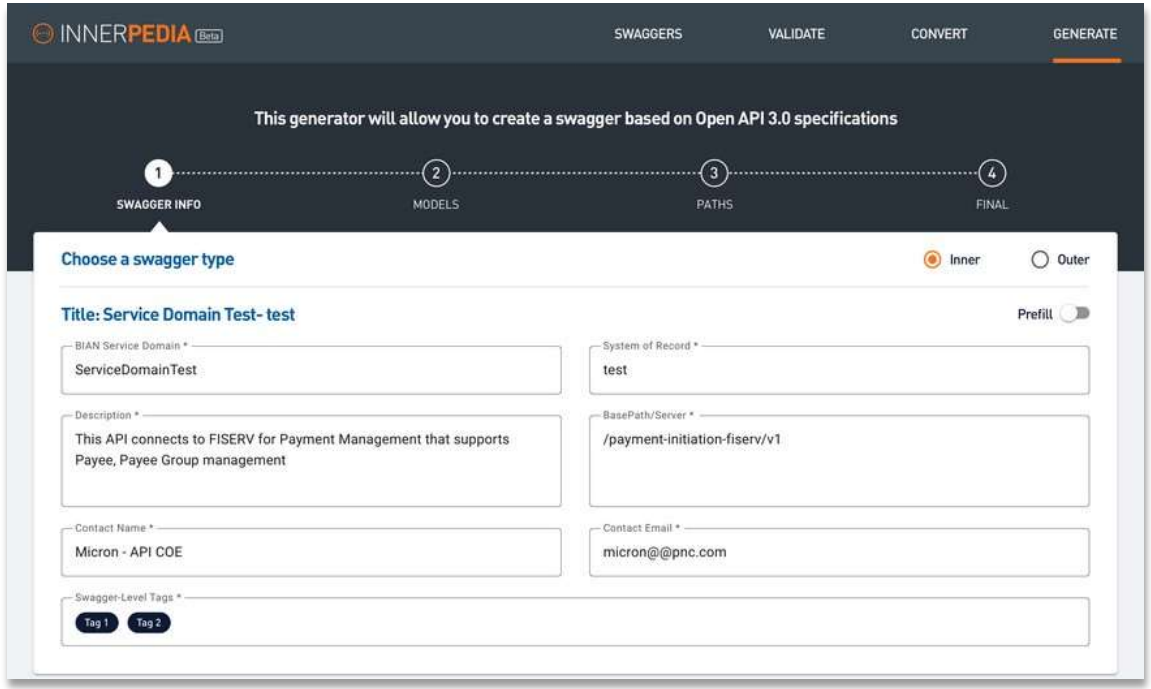
Descriptions

Descriptions of the service domain have been provided, as well as examples of use to provide specific use cases for each microservice

Swagger Generator & Validator

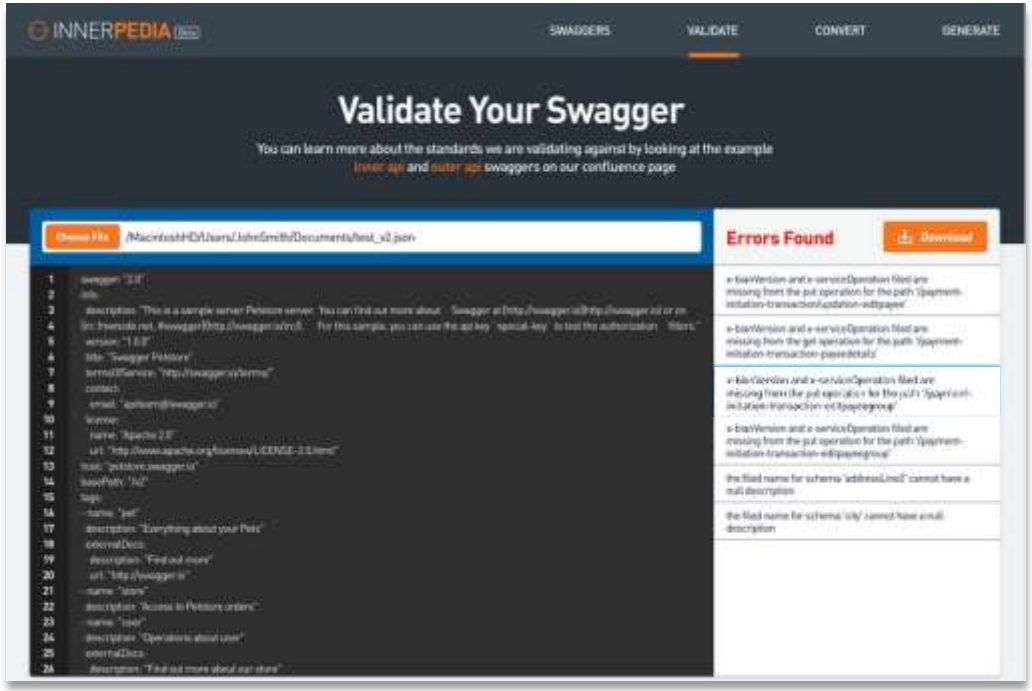
The Swagger Generator and Validator were built to help users model their services uniformly and in adherence with PNC and BIAN standards

Swagger Generator



- Mechanism for users to build swaggers (contracts) in accordance with PNC standards
- Includes stored BIAN and API model data so that fields auto-populate and restrict the user from deviating from modeling guidelines

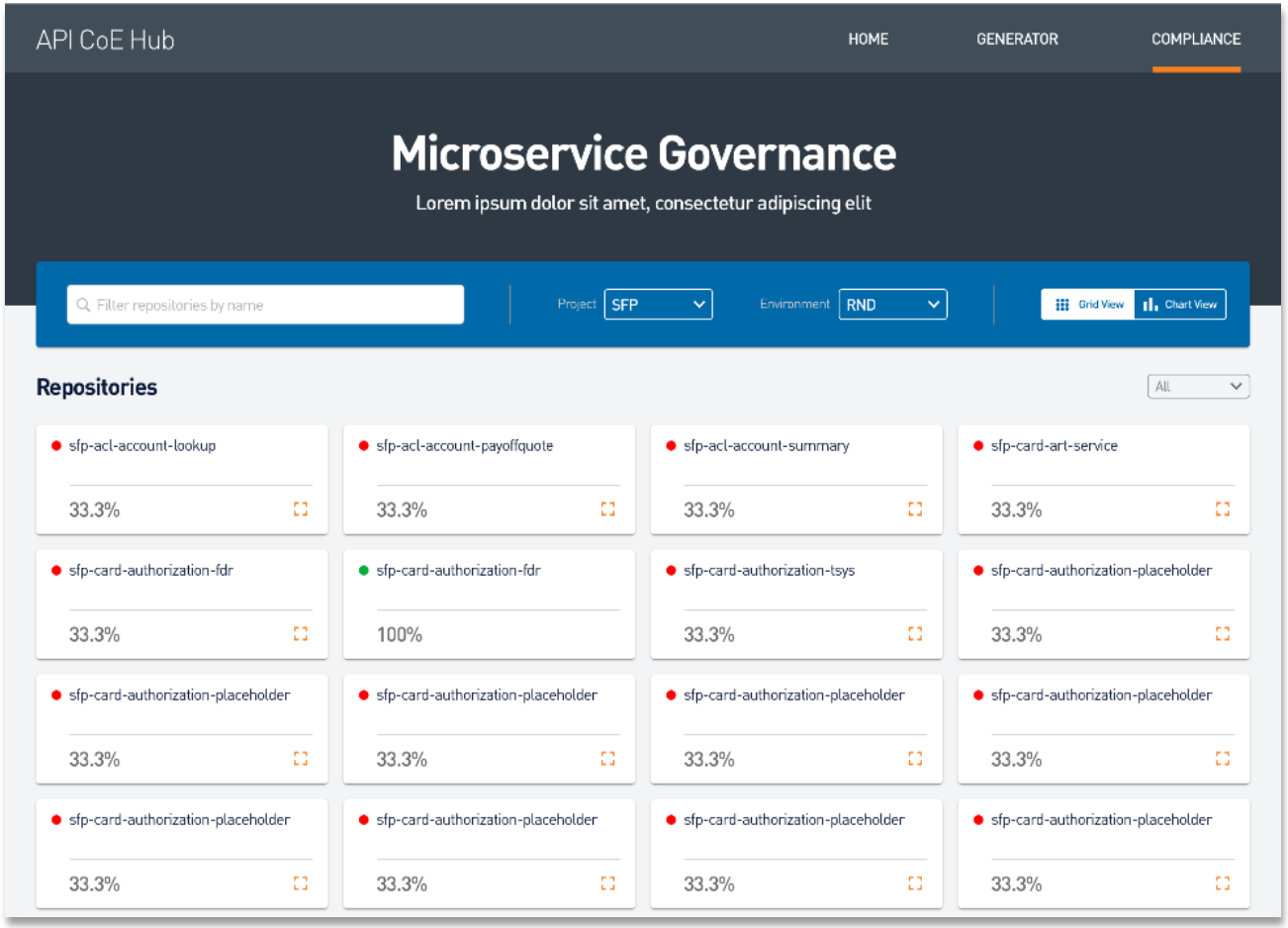
Swagger Validator



- The validator provides users the opportunity to check that their swagger is compliant, providing feedback for any errors
- The tool checks to ensure naming conventions, BIAN standards, and other modeling rules are adhered to

Compliance Dashboard

The compliance dashboard allows both development teams and managers to monitor adherence to modeling and swagger standards



- The compliance dashboard displays compliance checks that are performed when applications are built and deployed
- Provides a mechanism to check for adherence to specific standards, like swagger and data modeling compliance

PNC AND BIAN PARTNERSHIP

BIAN-Inspired Innovation & Partnership

The API Toolkit, a combination of the Endpoint Catalog and API Model, is a key enabler in ensuring BIAN standards are adhered to at PNC and provides PNC a mechanism to enhance the BIAN industry framework

Endpoint Catalog

Universe of all BIAN-inspired endpoints mapped to service domains and their corresponding SORs

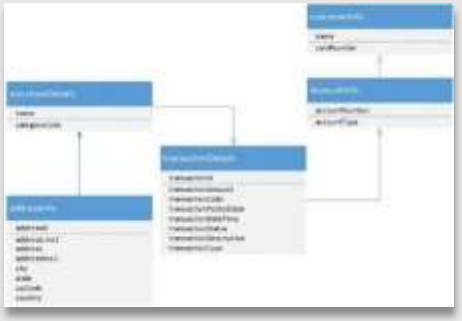
Endpoint Catalog

- Ensures all endpoints developed leverage BIAN standards
- Identifies areas to extend BIAN endpoints to meet PNC requirements

API Model

A collection of ER models that details entity and property relationships for each service domain, depicted in logical ER diagrams and provided to users in a JSON format

ER Model



- Provides modeled out entities and properties for each BIAN service domain
- Allows for automated population of swaggers to ensure standardized modeling and naming conventions are used across development teams

Extending the Model | PNC and BIAN Partnership

The work and analysis conducted to develop the Endpoint Catalog and API Model provide a mechanism for PNC to contribute to the BIAN model by extending both the framework's endpoint list and Business Object Model (BOM)

Feedback Loop:

1. Leverage the existing BIAN endpoint list and customize them for PNC's use cases and transactions
2. Understand PNC SOR capabilities and map them to the BIAN model
3. Host working sessions to ensure PNC-inspired endpoints and models align to the BIAN framework
4. Extend the BIAN framework to meet PNC requirements
5. Contribute back the delta between PNC requirements and the existing model to help grow, enhance, and evolve the overall framework

Absa (&TCS)

Table of Contents

Context

About Absa

Absa is a truly African brand committed to finding local solutions for uniquely local challenges and everything they do is focused on adding value. To this end Absa offer all clients across the continent a range of retail, business, corporate and investment, and wealth management solutions as well as ensure a positive impact in all the countries where it operates in.

Vision : To become digital bank of choice.

Primary Drivers for BIAN Adoption

- The aim of the bank is to become the digital bank of choice across the African continent with customer centricity at the core. Absa wanted to achieve this by ensuring the faster launch of digital products and features for its customers. The bank had a vision to Lego-fy its banking services by repackaging services into reusable building blocks which would guarantee them success in their digital transformation journey.
- Decompose existing legacy middleware into API, Microservices.
- Introduce new age digital channels and establish DevOps, CI/CD for faster Time To Market.

Our new Microservices/API platform - Way Forward - Principles

The new platform is not a like-for-like replacement for existing ESB middleware

- Service will not be built to retrofit existing channels.
- Not just a protocol change i.e. SOAP to REST.

The new platform is more than a “gateway”, it provides :-

- Decoupling and abstraction from the core banking and other backend services.
- Standardization and governance via its use of the BIAN Framework.
- Orchestration where it is required.

Vendor applications and in-house developed solutions needs to comply with the BIAN-based Absa standards that have been adopted when competing for business.

Designed with a Microservices architectural style to :-

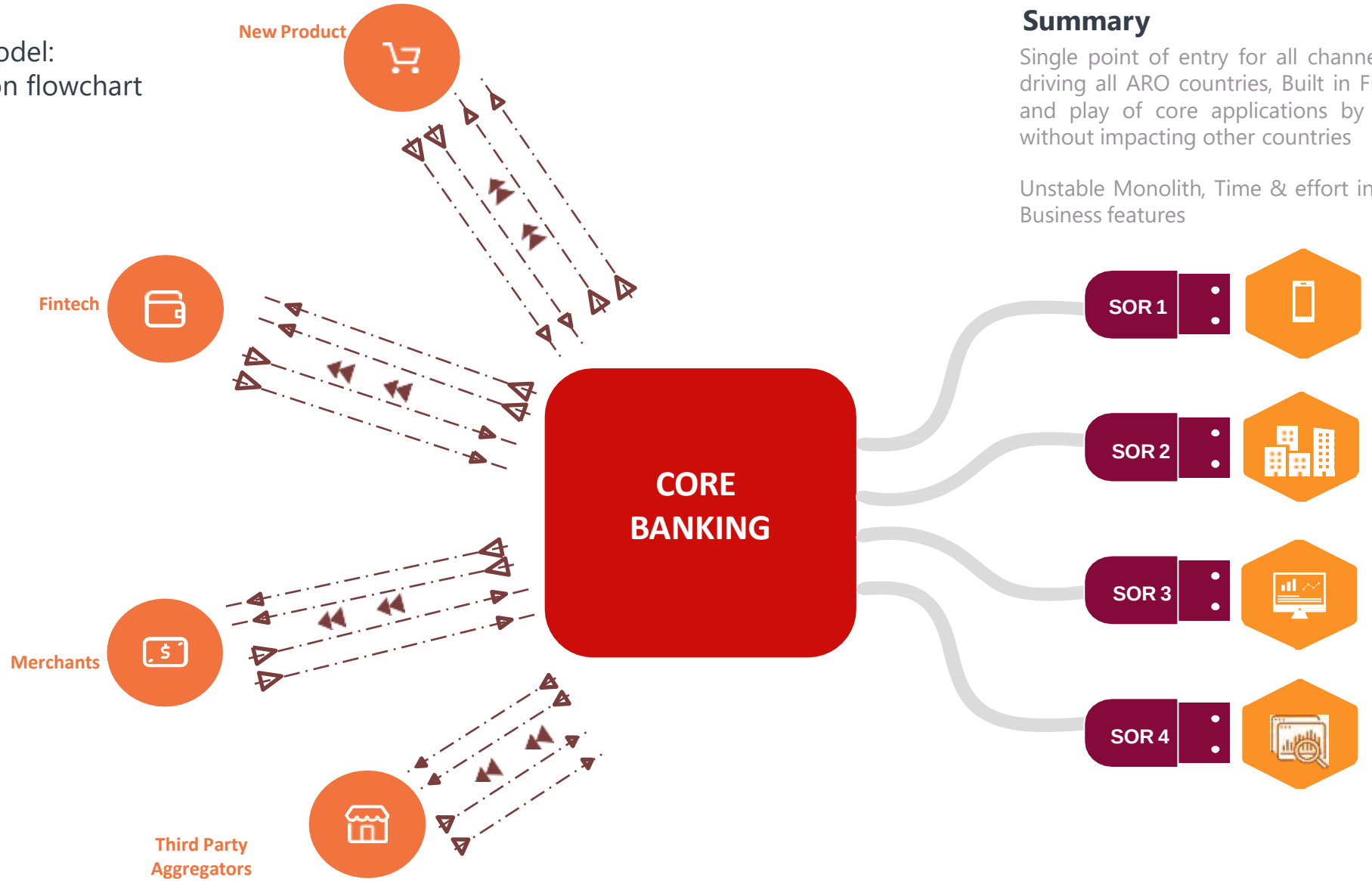
- Decouple complex systems from business functions like with SOA (Service-oriented Architecture) but with more service independence and more resilience.
- Leverage of AGILE and best practice principles coming out of our industry.

The new platform seeks to adopt a standardized approach by implementing ‘common’ architecture principles across the Absa Regional Operations landscape.

The new platform is designed to decouple back-end solutions so that that a “plug-n-play” approach can be adopted to replacing back-end applications.

Existing Architecture

Core banking model:
Single transaction flowchart



Summary

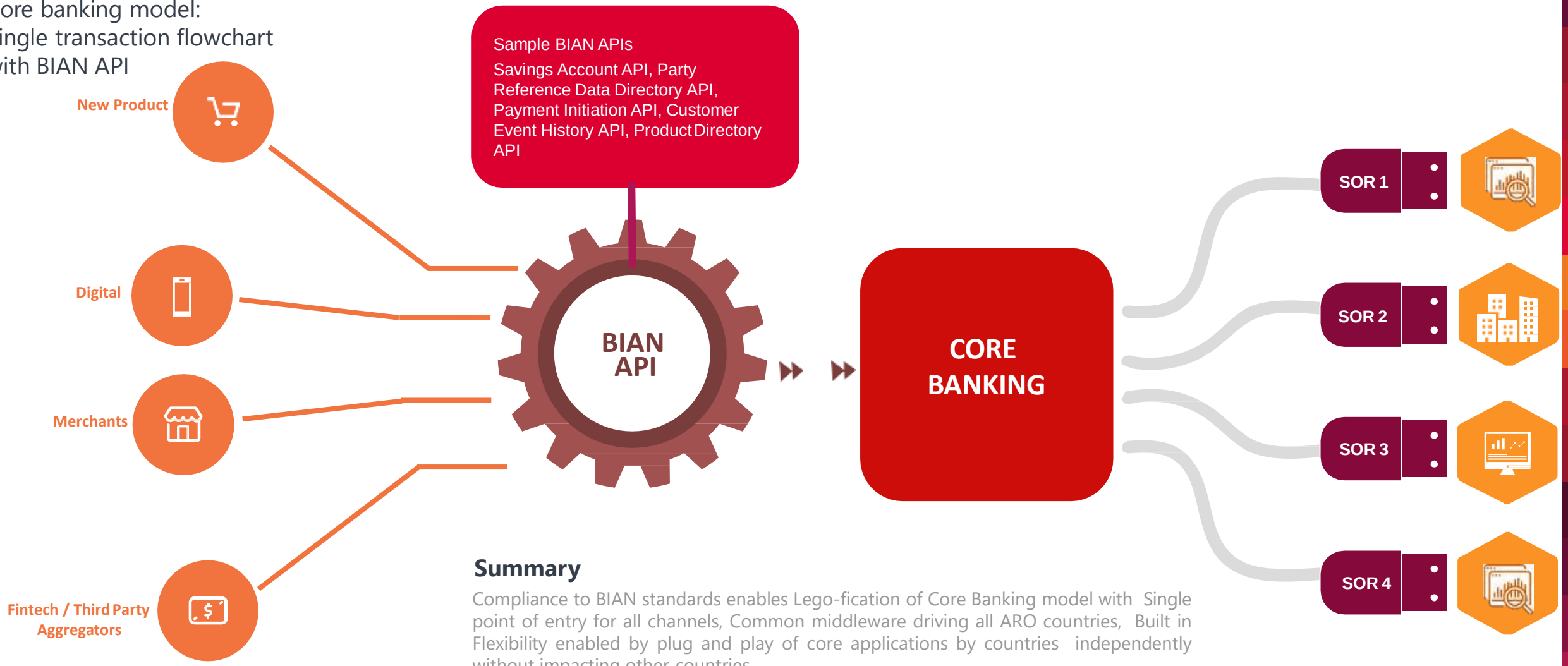
Single point of entry for all channels, Common middleware driving all ARO countries, Built in Flexibility enabled by plug and play of core applications by countries independently without impacting other countries

Unstable Monolith, Time & effort intensive to introduce new Business features

Applications | Core Banking | ARO Apps | Multiple Integrations with Core banking | Integrations through multiple languages | Core Banking Integrating with internal applications

BIAN Adopted Architecture

Core banking model:
Single transaction flowchart with BIAN API



Summary

Compliance to BIAN standards enables Lego-fication of Core Banking model with Single point of entry for all channels, Common middleware driving all ARO countries, Built in Flexibility enabled by plug and play of core applications by countries independently without impacting other countries

Secure, Reusable, Standardized enabling cost and effort efficient introduction of Business features

Applications | Core Banking | ARO Apps | SINGLE Integrations with Core banking using BIAN API | Integrations through SINGLE BIAN | Core Banking Integrating with internal applications

Program Approach

MVP – Current Approach

•Focused on onboarded channel only

Pros:

- Faster TAT (Turn around time) for channels
- Zero wastage

Cons:

- Additional cost due to continuous engagement with SORs and Channels
- Lack of reusability. Need enhancement every time

Big Bang

•Consider all channel needs at once

Pros:

- Zero rework as all requirements analysis completed before start of development

Cons:

- Can lead to wastage due to un-used APIs
- Increase in time to market of new channels
- Disconnect between business needs and solution offered
- Not able to gain competitive advantage

Hybrid – Suggested Approach

- Consider requirements for existing key channels and uplift APIs for new requirements on need basis

Pros:

- Build right first time
- Zero or minimal re-work
- Reduce cost for SMEs support
- Faster Time to market of channels introduction

Cons:

- None

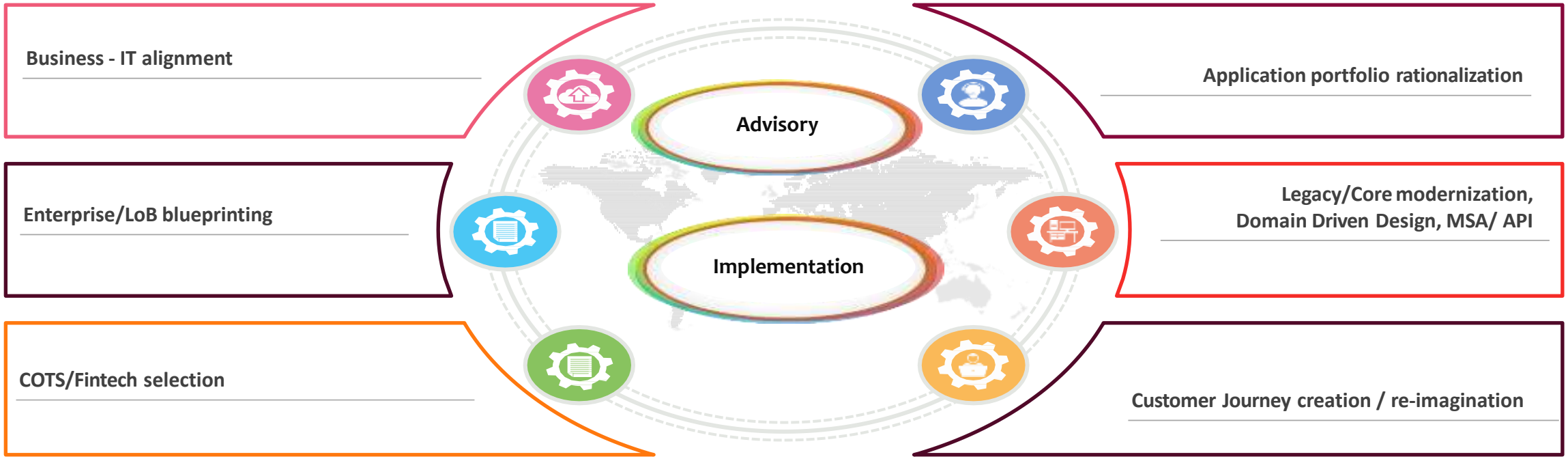
- Adopted green field implementation approach to maximize benefits of agile architecture.
- Targeted new Channels to be onboarded first on the new platform followed by transformations of existing channels.
- MVP approach followed for quick onboarding of transactional Chat Banking channel and now we are looking at analysing existing capabilities of all digital channels.

BIAN Adoption trends & recent experiences

- Multiple customers
- Few customers
- Very Few customers

Speed to Market – Launch products, features, capability faster

- **Business - IT Alignment & Enterprise/ LoB Blueprinting** – Multiple customers
- **Application portfolio rationalization to identify gaps, overlaps and misaligned applications** – Multiple customers
- **Match and select vendor solutions/COTS/Fintech for business capabilities** – Very Few customers
- **During M&A process to select competing capabilities from the merged organization** – Very Few customers
- **Moving away from ESBs to Standardized APIs and Microservices on Cloud** – Multiple customers
- **Legacy modernization / Core systems repurposing and moving towards coreless banking** – Multiple customers
- **Create DDD based right sized services for any greenfield / brownfield implementations** – Multiple customers
- **Innovation - Customer journey creation / re- imagination** – Few customers



The Journey – We started with an execution approach …

Post PoC Implementation, a detailed execution approach was put together for the entire program

TCS BIAN Assets | Templates | Enablers | Best Practices

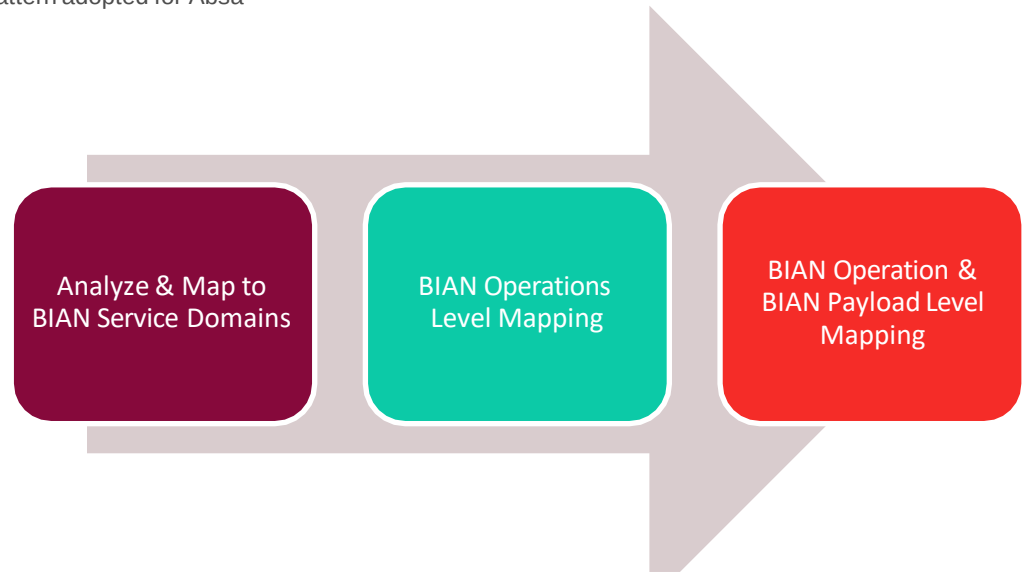
	Pre- work	Analyse	Align & Adopt
Activities	• Understand back end applications & services in scope • Identify stakeholders – All SMEs and reviewers needed for the project. • Confirm level of BIAN API adoption. • Agree on a timeline for execution.	• Understand the business processes in scope- entire process flow from channel to SORs. • Collaborate with stakeholders for thorough analysis. • Review Understanding	• For BIAN aligned DDD services, create Service URIs & Contracts. • Present to Stakeholders for review and signoff
Deliverable	• Detailed proposal (scope, execution plan, teams involved and timeline)	• Mapping of existing services to DDD based BIAN service Domains and create service catalog.	• BIAN based service specifications along with URIs and contracts .

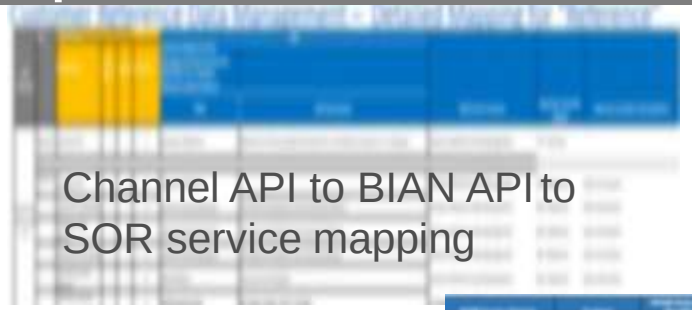
Identified the right API Adoption pattern & delivered artefacts…

API adoption pattern	Align Service Domain & URI	Align Service Domain, URI and Control record	* Align Service Domain, URI and Control record as well as BIAN BOM
Impact	Consumers will have to be made aware of changes in the URI.	Request response will have to be changed, mapping dictionary to be maintained for data elements.	Entire data model has to be aligned with BIAN.

Increasing Level of alignment

* Pattern adopted for Absa



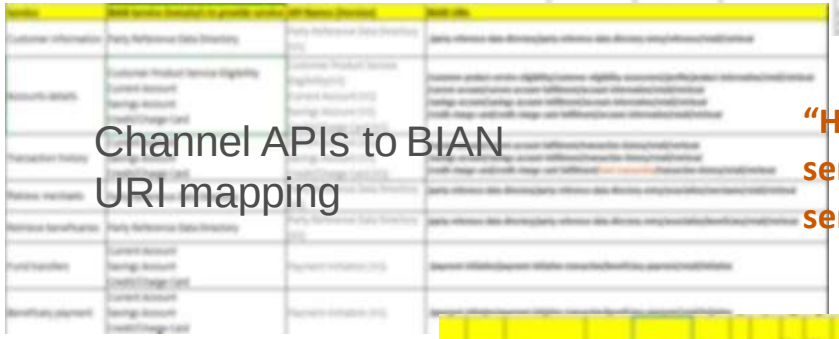


Channel API to BIAN API to SOR service mapping

"Helped consolidate multiple services providing same capabilities. Eg. 20 payment services were consolidated to 4"

"Helped identify reuse across channels "

Mapping multiple channels to BIAN API



Channel APIs to BIAN URI mapping

"Helped create future ready services, channel agnostic services "

"Helped provide the right level of abstraction thereby ensuring SOR changes are not impacting the channels."

BIAN APIs & request response mapping to SOR services request / response

Handling few challenges faced during adoption….

BIAN provided Payment initiation semantic API has this URI for initiation operation-
/payment-initiation/{sd-reference-id}/payment-initiation-transaction/initiation

While implementing for Absa we added multiple sub qualifiers for the various business scenarios as shown
/v1/payment-initiation/payment-initiation-transaction/ fund-transfers/domestic/initiation
/v1/payment-initiation/payment-initiation-transaction/ fund-transfers/ nternational/initiation
/v1/payment-initiation/payment-initiation-transaction/ bill-payments/initiation

Challenges

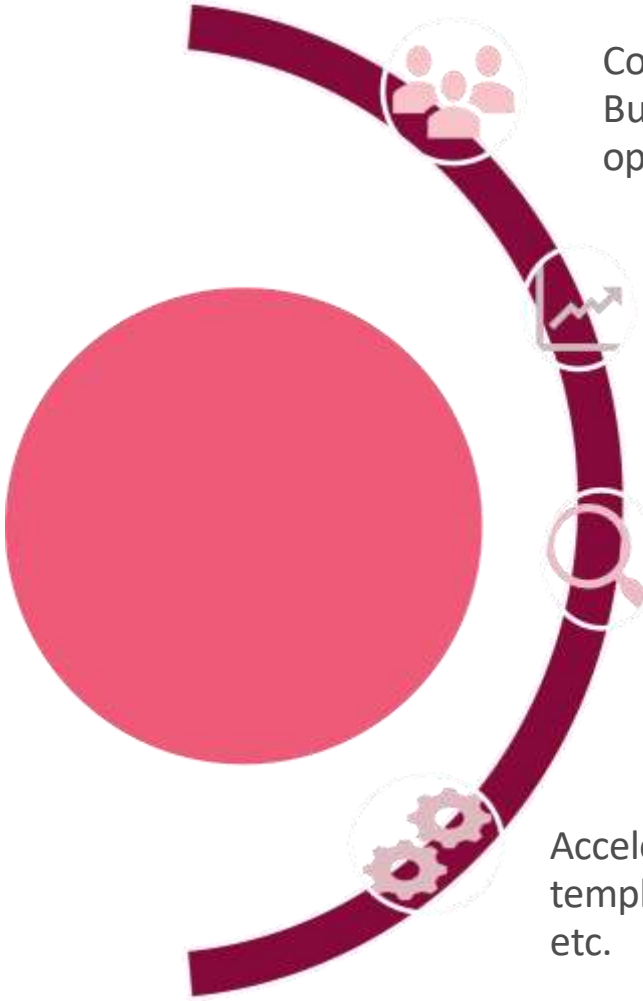
- BIAN semantic APIs will serve as reference APIs hence during implementation journey, for all the needs specific to banks we need to introduce the right sub-qualifiers in the service operations so that they are discrete and non-overlapping and rightly mirror the banks context.
- Most of the Bank's data elements were mapped to BIAN BOM, for the ones which were not available in BIAN BOM were added and Bank's own custom data model extending BIAN BOM was created. This exercise needs a lot of domain expertise as well as in-depth knowledge of BIAN BOM. It is quiet time consuming, so we have created a framework to automate some aspects of this data mapping exercise.
- Synchronizing with latest BIAN releases have to follow a well -defined process so that there is no impact to execution timeline and also ensures the latest BIAN updates are incorporated.

Sample updates in BIAN version 9

Service Domain Name Changes

- Fraud AML/Resolution => Fraud Resolution
- Credit/Charge Card => Credit Card
- Customer Product/Service Eligibility => Customer Product and Service Eligibility
- Document Services => Document Library
- Party Data Management => Legal Entity Directory
- Customer Reference Data Management => Party Reference Data Directory
- Contact Dialogue => Session Dialogue

Critical success factors that helped ....



Collaboration with all stakeholders like Business team, technology team, operations team and EA team.

Roadmap & right execution approach –start with PoC or MVP for single LoB or entire LoB before scaling at a larger level.

BIAN implementation requires a thorough knowledge of BIAN framework so we deploy a team with prior experience that regularly connects with BIAN.

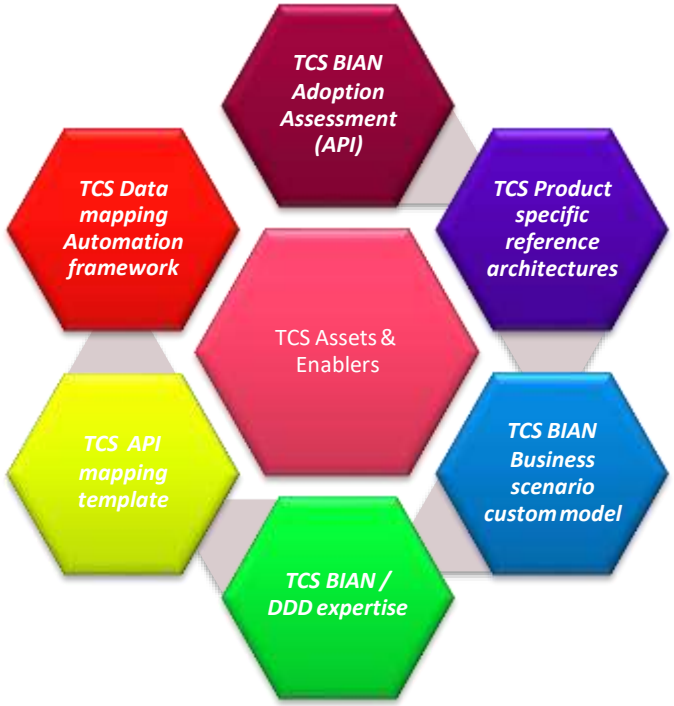
Accelerate adoption by usage of templates, assets, automation frameworks etc.

Sample feedback shared with BIAN

For a business scenario, we have a situation to handle Memo for transaction processing within a Current/Savings account service domain. We have not come across memo handling capability. Can it be added?

In BQ Association instance record of Party reference Data directory SD, we have 'Proxy/Representative/Power of Attorney Reference' element to capture associated reference details . Can we extend it to capture address, email etc.

We need a field in credit card control record to store reference to 3rd Party who acquired this customer. Can this be added?



Delivered the below benefits to the bank…

Speed To Market

- Quicker Integration due to standardization.
- Plug & play integration capabilities.
- Avoidance of Vendor –lock in.
- API catalog re-used across the enterprise leading to quicker development
- CI/CD to improve TAT
- Enabled API Marketplace business model

Increased revenue

- Capability to leverage ecosystem players leading to new revenue stream
- Seamlessly integrate with fintech and partners to deliver new features & products

Standardization & Scalability

- Standardization of architecture capability definition & information model,
- Leaner application stack, Optimized calls to the core
- Discrete Business functionalities driven by Modular service domain design
- BIAN aligned APIs with clarity of purpose
- Scalability and availability as per business needs
- Enabled Cloud readiness

Cost reduction & Reuse

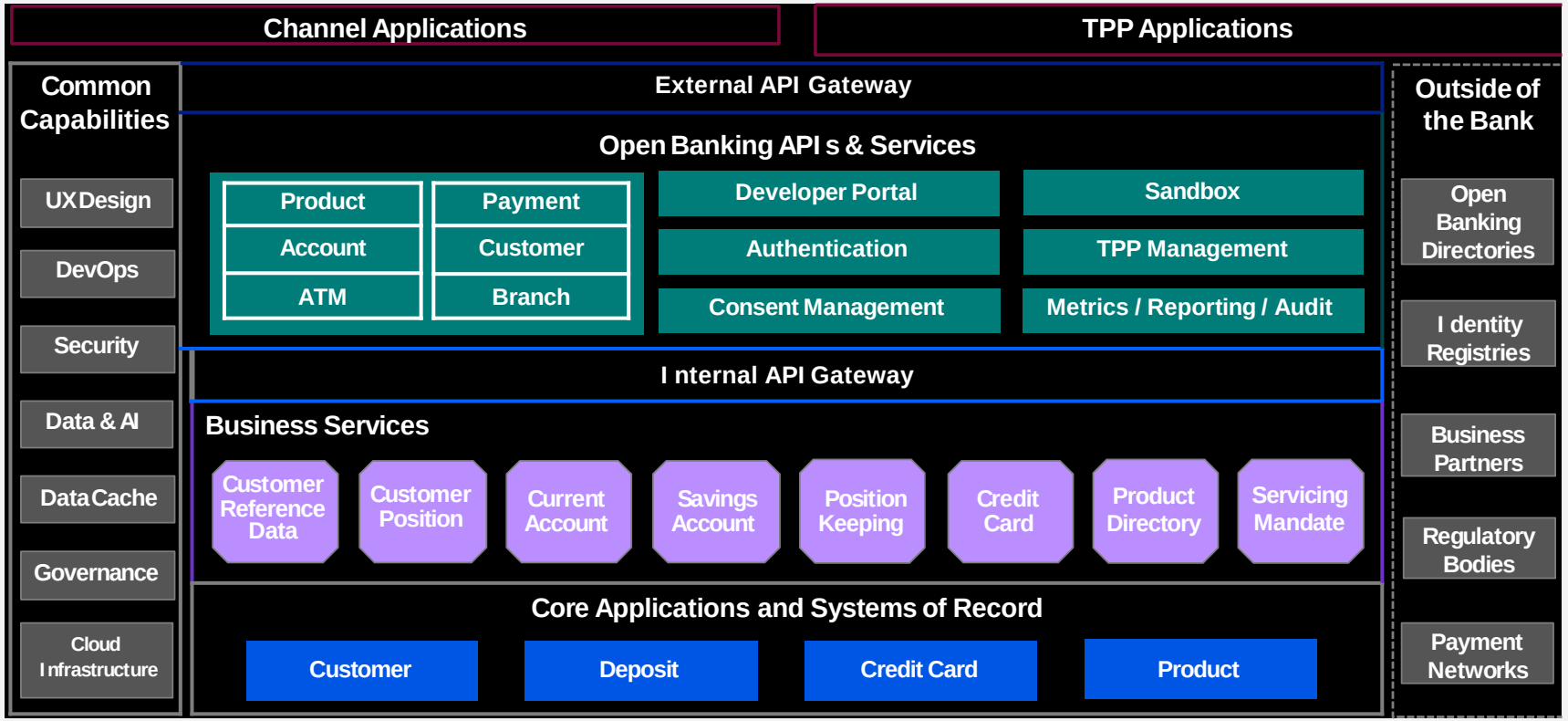
- API re-use across the enterprise.
- Cost reduction due to reduced API footprint.

Customer satisfaction

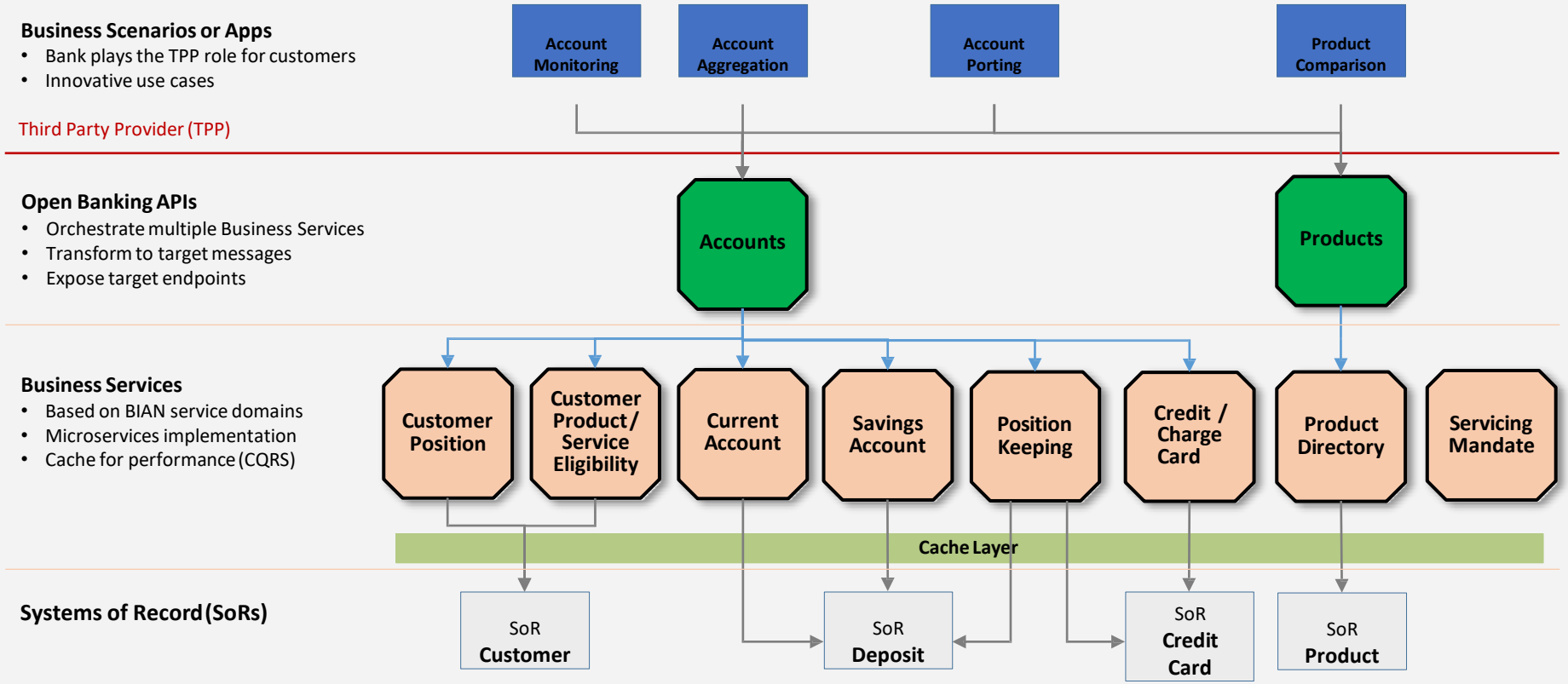
- Improved customer satisfaction due to quicker launch of products.

BIAN

Open Banking Reference Architecture



Open Banking APIs – Products and Accounts

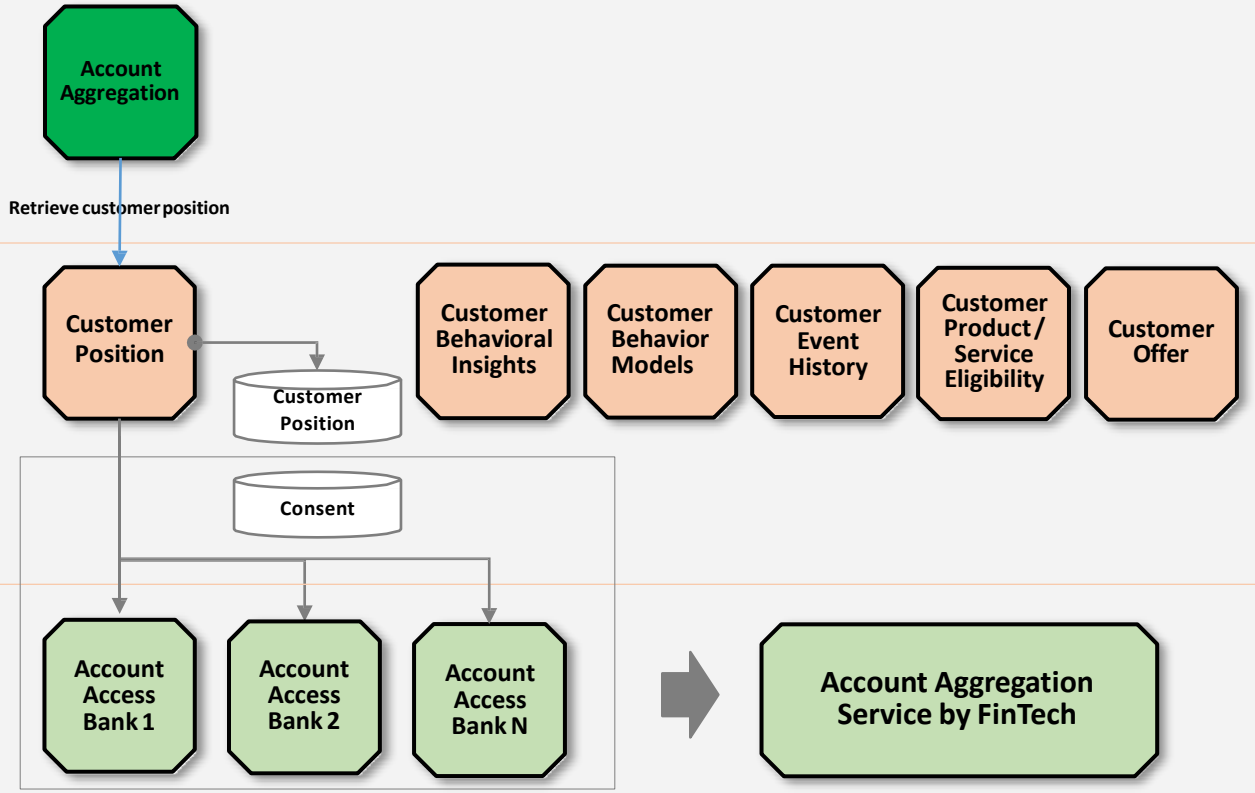


Account Aggregation – Better Customer Insights

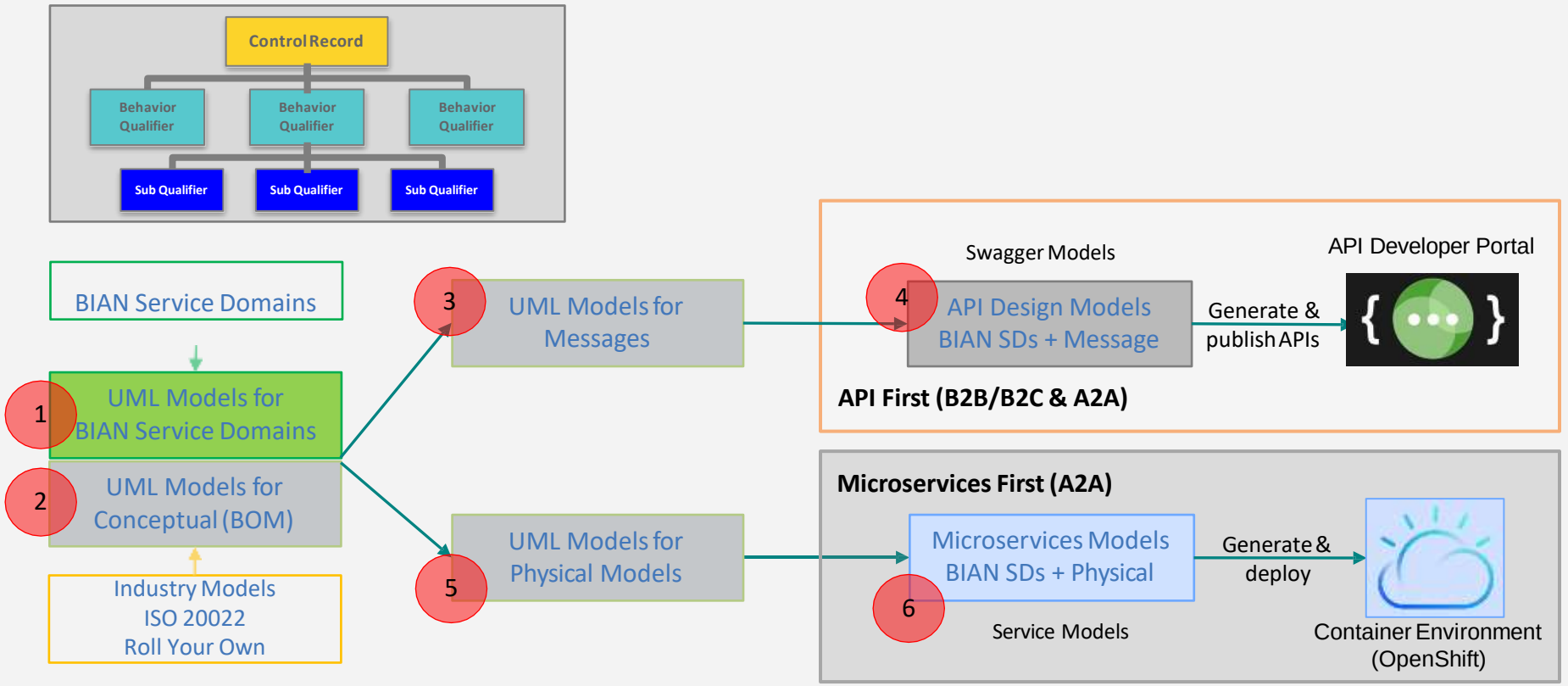
Account Aggregation Service & API
- Customer position aggregated from multiple financial institutions
- Used by channel applications

BIAN Based Business Services
- Customer Position maintains …
- Customer insights from analytical models
- Offers to customer

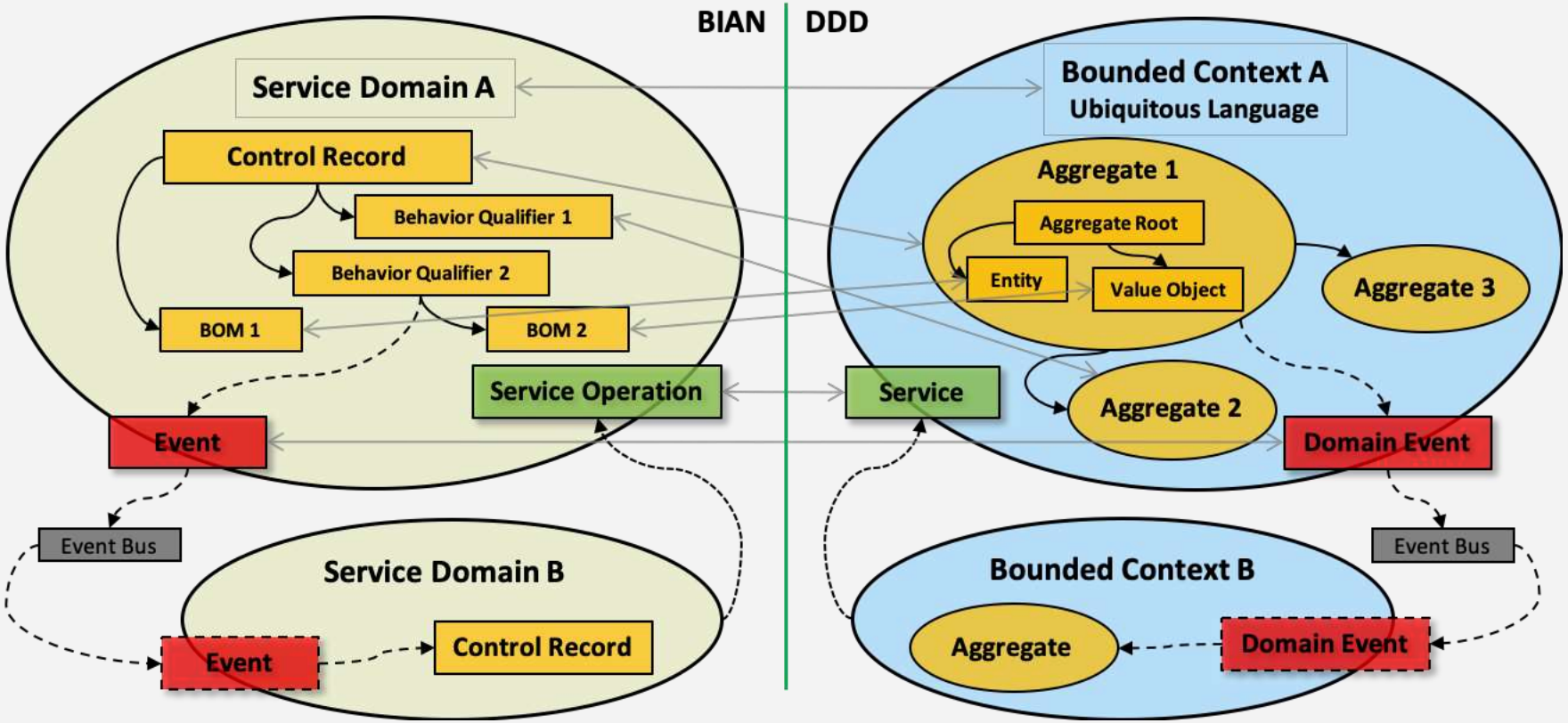
Account Access to other banks
- Account Access to multiple banks
- Using FinTech offering to simply the integration



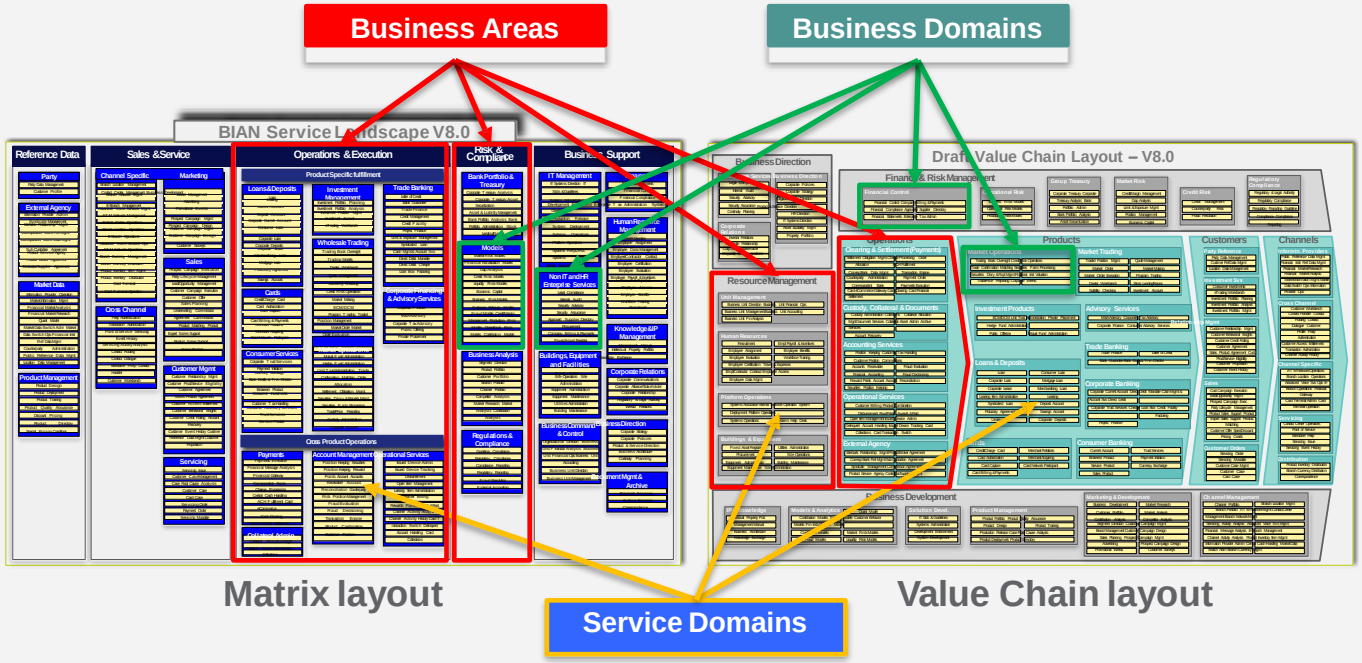
BIAN Based APIs and Microservices



BIAN and Domain Driven Design (DDD)



BIAN Service Landscape – Matrix vs. Value Chain Layout

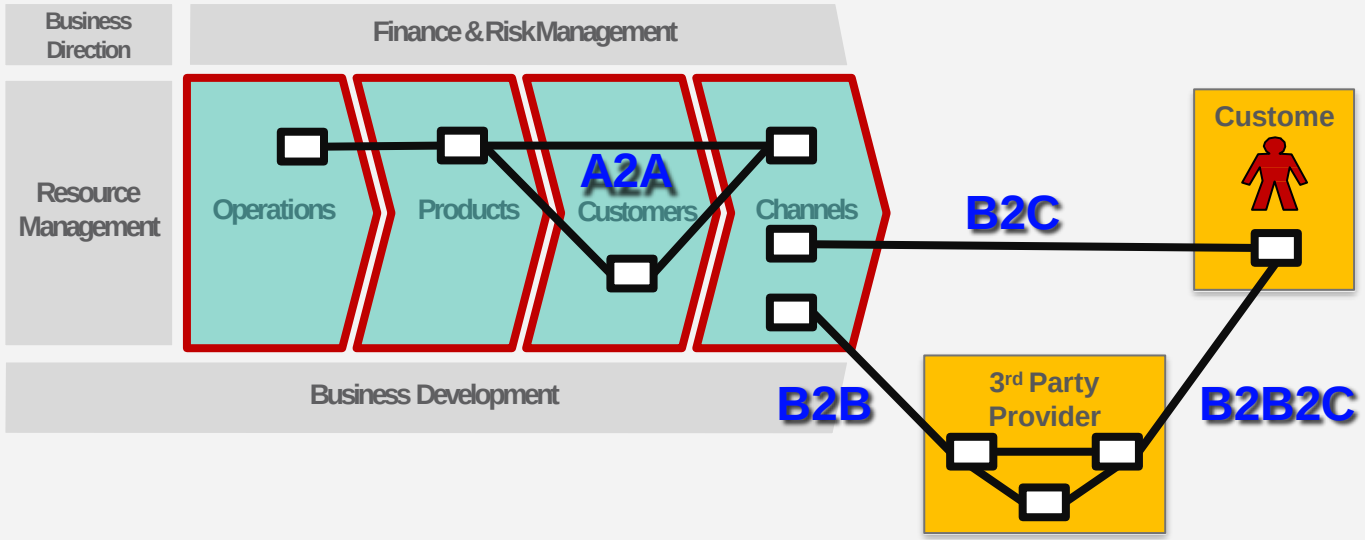


BIAN Value Chain Landscape and Scopes of APIs

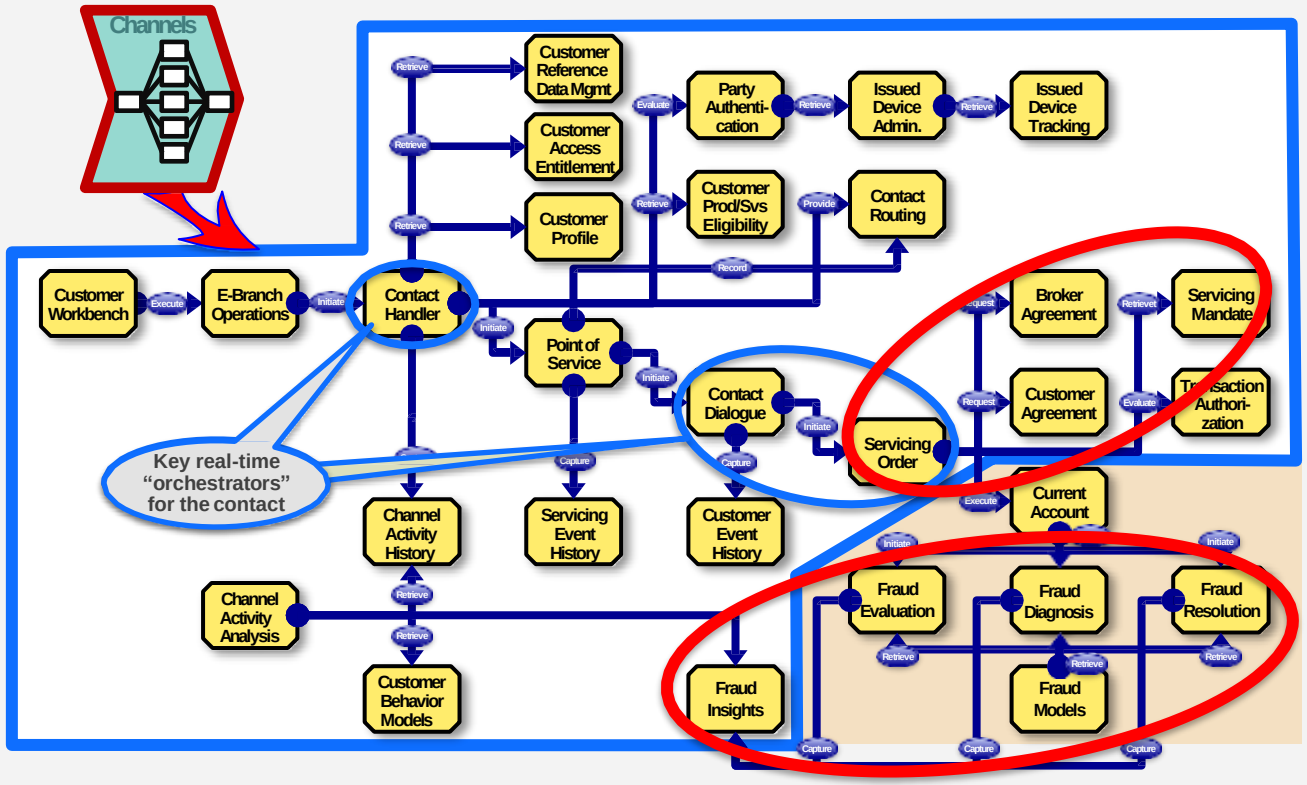
The BIAN Service Landscape V8.0

Business Areas:

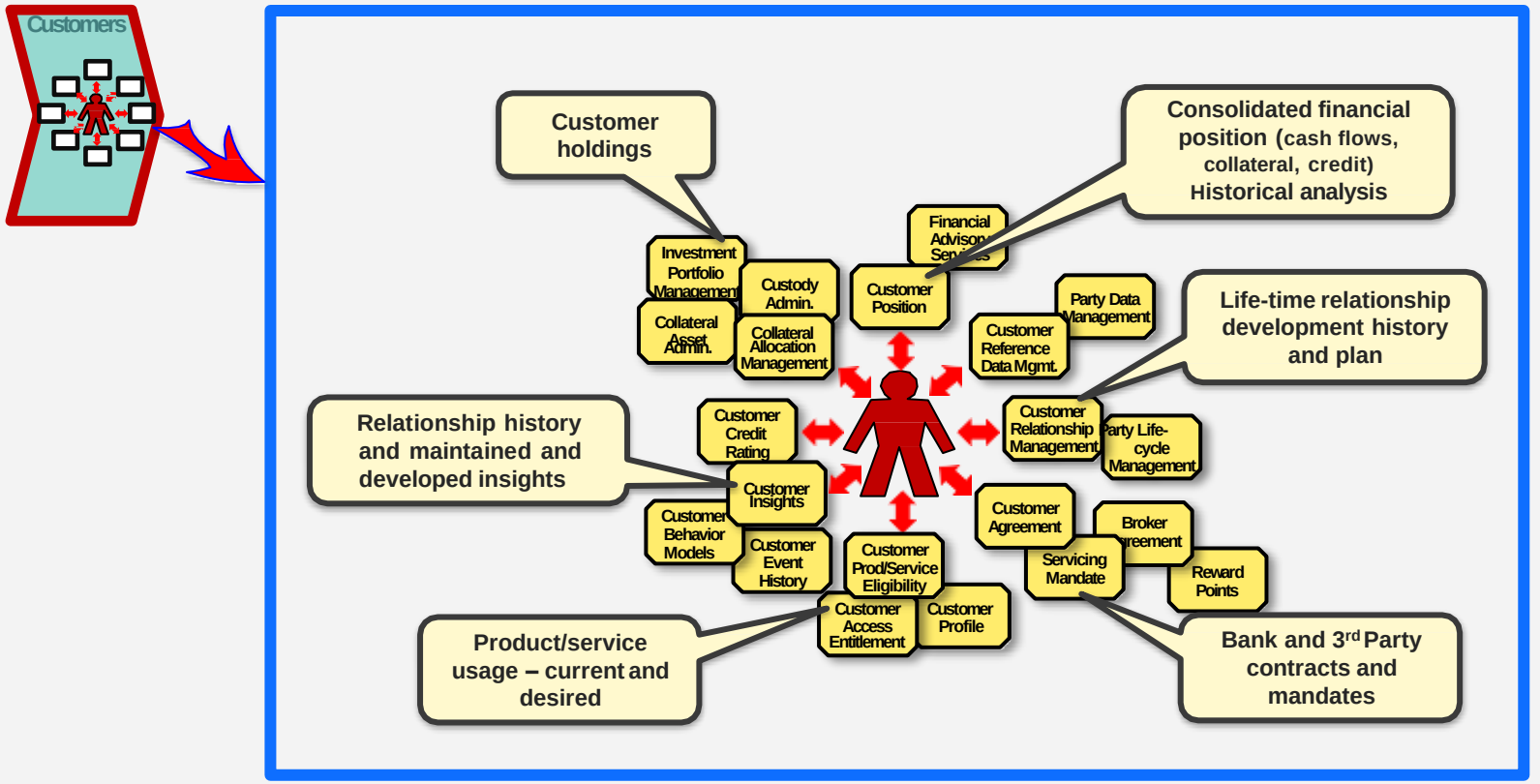
- **Operations** – cross-product operations
- **Products** – synergistic product / service combinations
- **Customers** – integrating customer insights & preferences
- **Channels** - governing access – both at the contact & servicing mandate / relationship levels



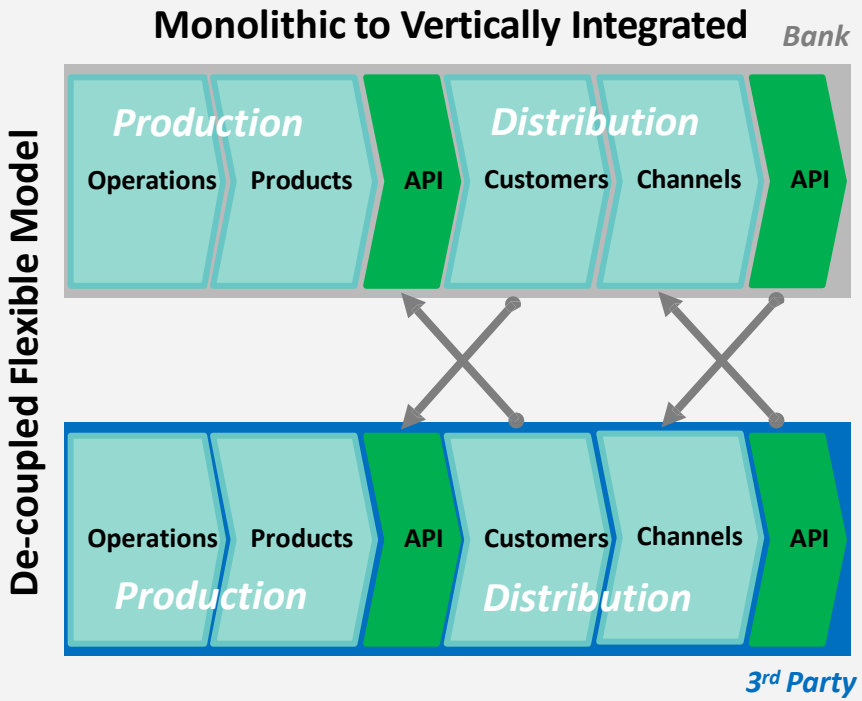
Channels – External Access Control

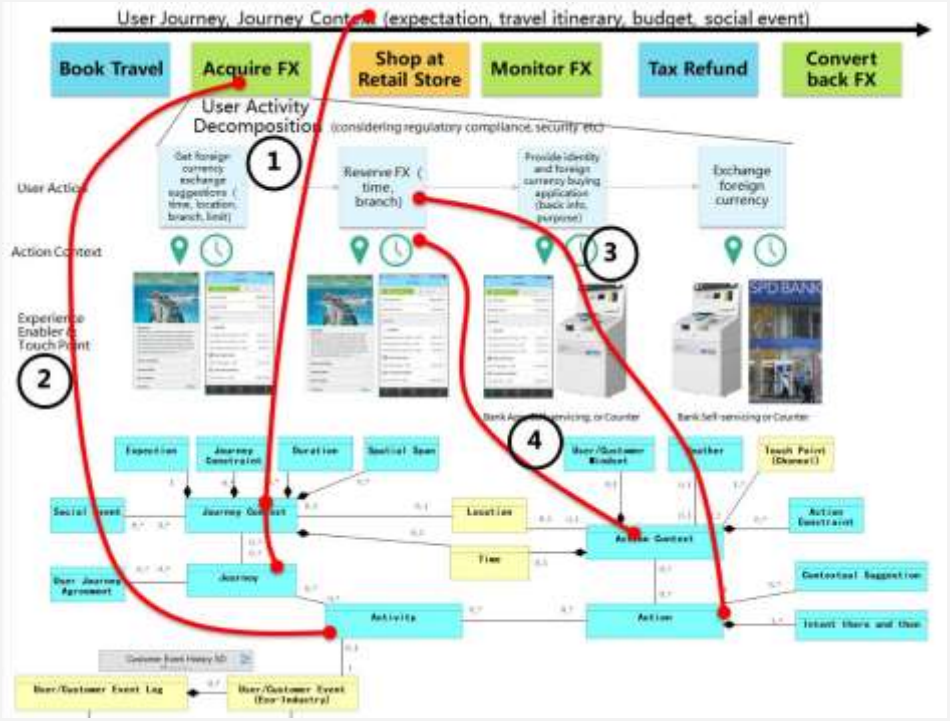


Customers – Positions and Insights



Value Chain Decoupling and Integration – New Business Models



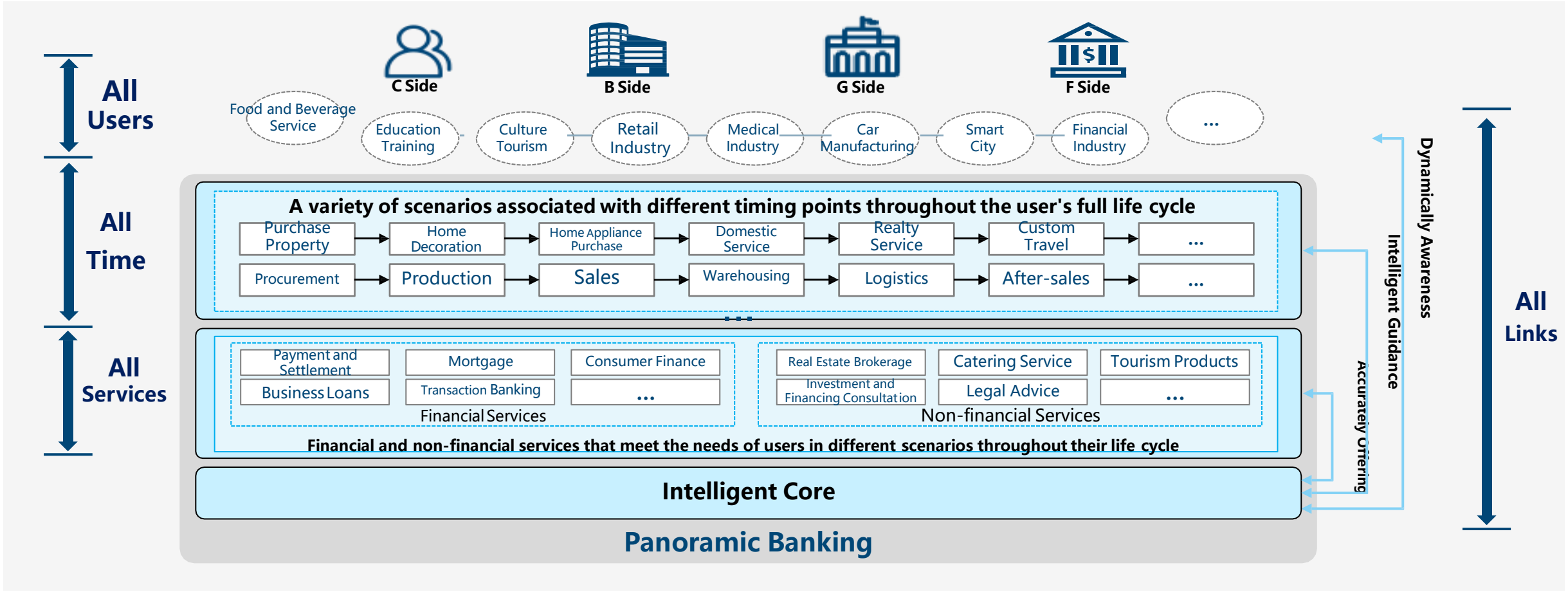


Contents

Panoramic Banking Concept and Vision

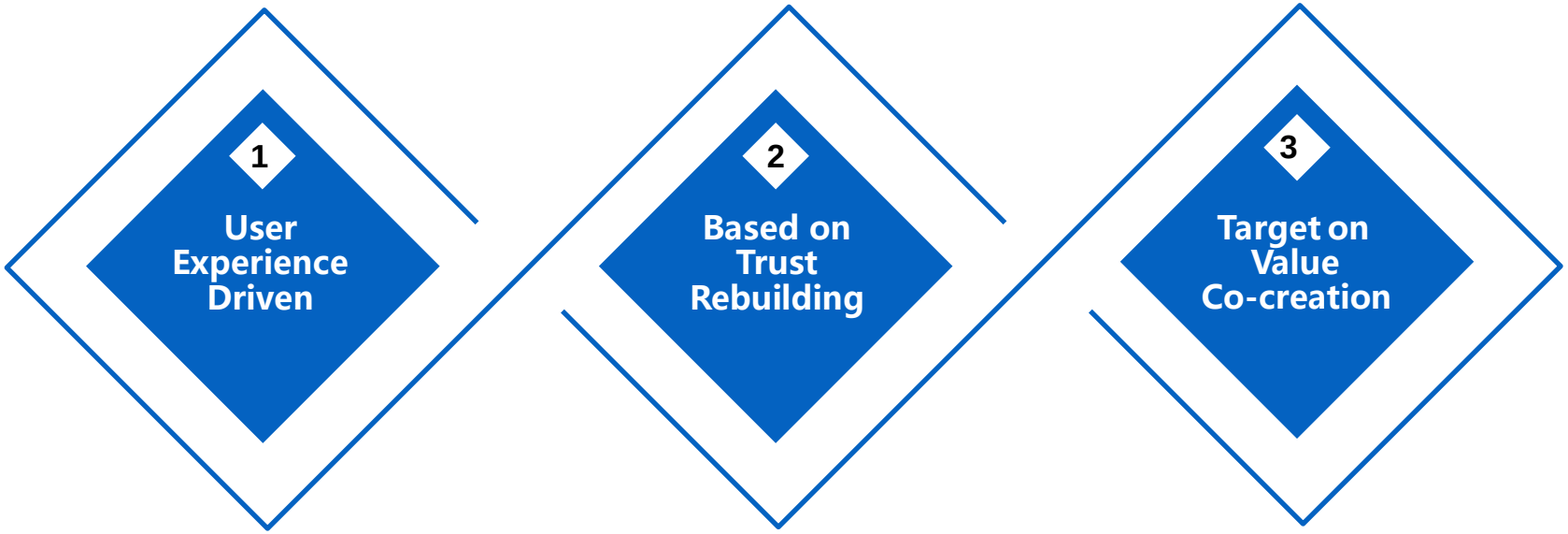
Panorama banking means that bank builds platform-based ecosystem around the needs of customer life cycle and production cycle, sharing brand, channel, customer, technology and other resources with business partners, uses data intelligence dynamically perceiving customer requirements, provids integrated financial and non-financial products and services, so as to create value adding and achieve best experience for customers

Facing "all users", running through "all time", providing "all services" and realizing "all links"



浦发银行 SPD BANK

Panoramic Banking Features



- Focus on understanding and solving user’s pain points
- Integrate internal and external resources to meet user needs, provide quality services and improve user experience

- Improve the overall digital level of related parties
- Build digital credit system, reduce the information asymmetry between customer and financial institution and build trust of them

- Open capabilities, enable ecosystem partners achieve win-win cooperation
- Aims on value adding for users and partners, instead of playing zero-sum game

Panoramic Banking Basic Value Point

Traditional Banking

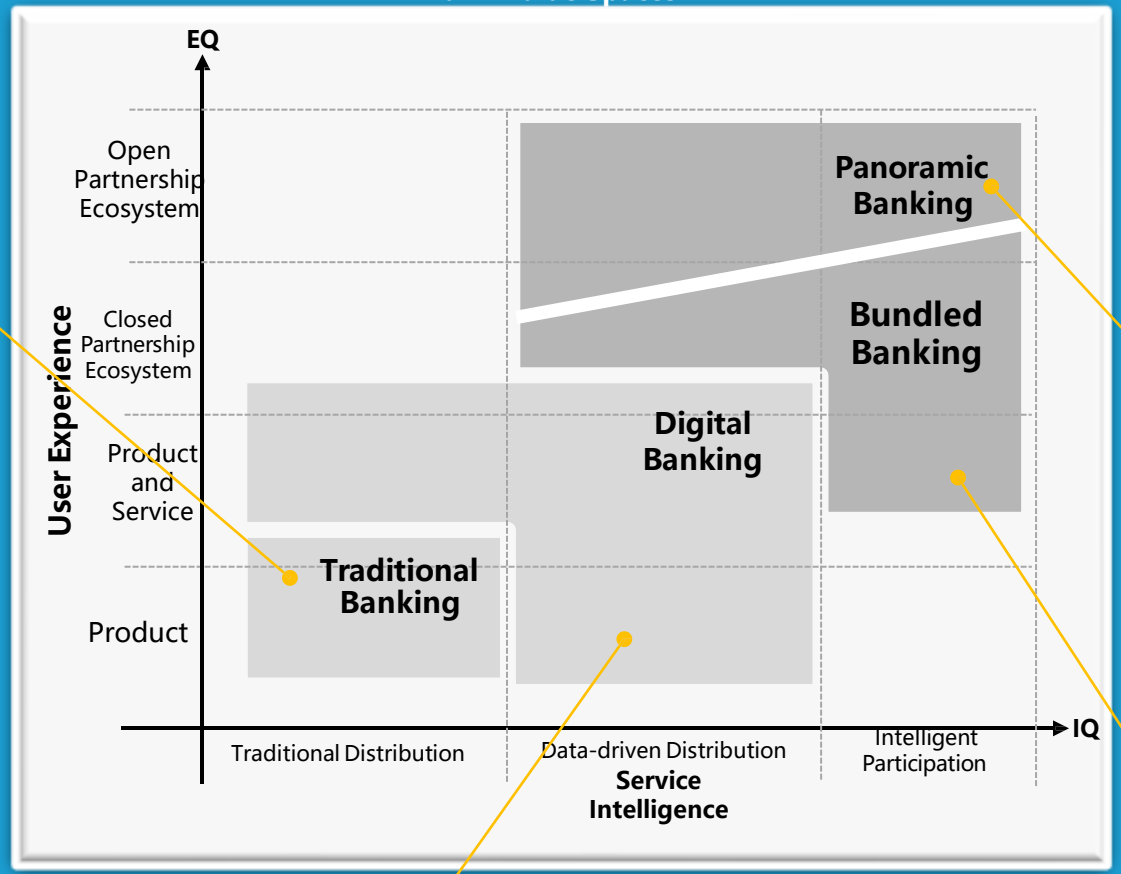
- **Realized value: Customers buy financial products or make financial transactions**
- The bank sells products or services to customers as an independent distribution channel. Such banks typically have high cost/revenue ratios and are prone to become infrastructure if lacking scale advantages

Digital Banking

- **Realized value: Customers buy financial products or make financial transactions**
- Most banks are in the process of transforming from traditional banking to digital banking. They use digitalization to upgrade traditional business model and enhance user experience. Banks in this range are still product distribution channels, profits have begun to shrink

Business Remodelling Grid

According to the combination of IQ and EQ, business remodeling grid has four main "value Spaces"



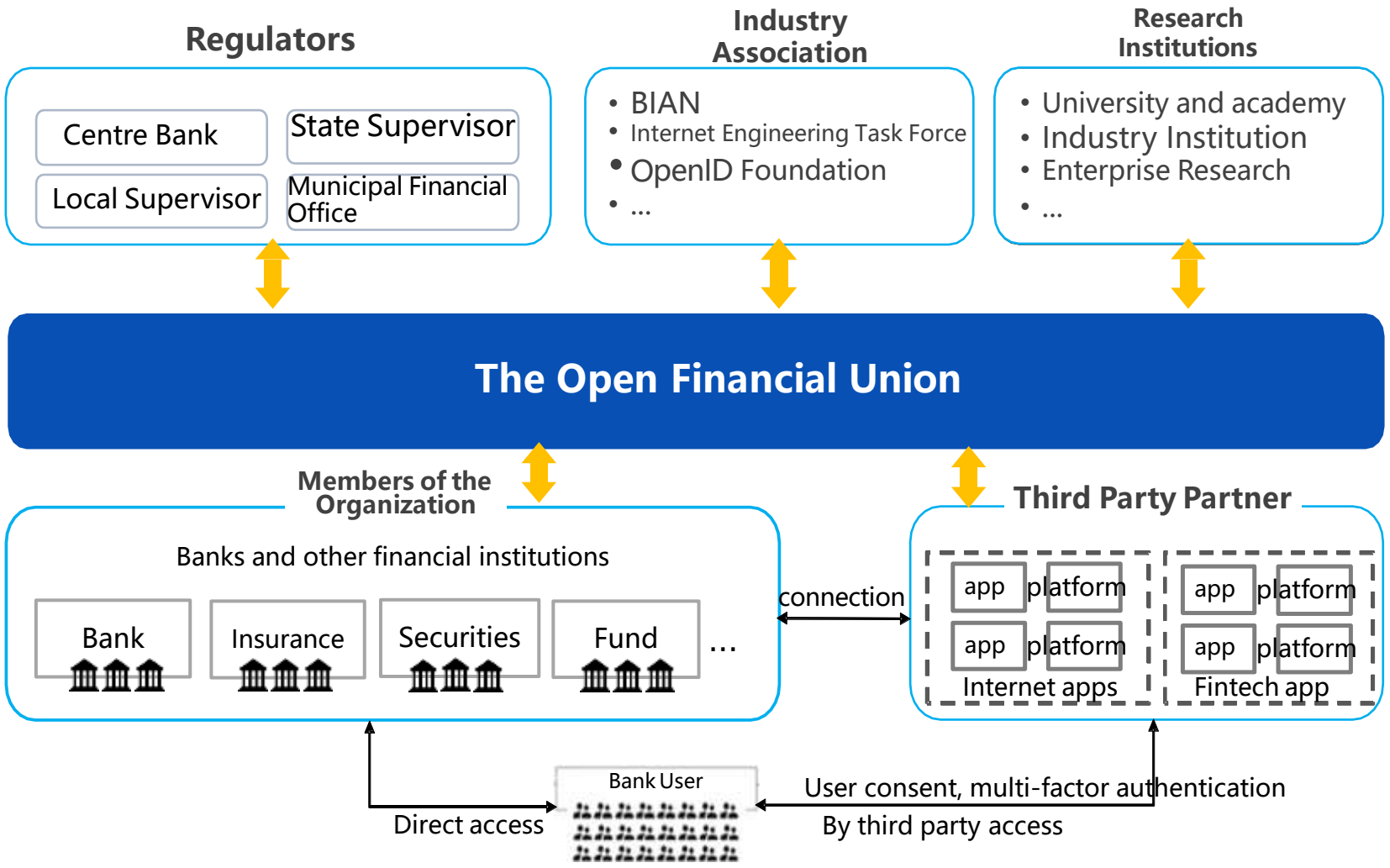
Panoramic Banking

- **Realized value: Customers production and consume life cycle serving**
- Start from customer needs, constructs financial and non-financial scenarios to support ecosystem parties to interact. Reduce the information asymmetry of parties and promote win-win cooperation. Provide intelligent services, enable effic6ient interaction and improves profitability

Bundled Banking

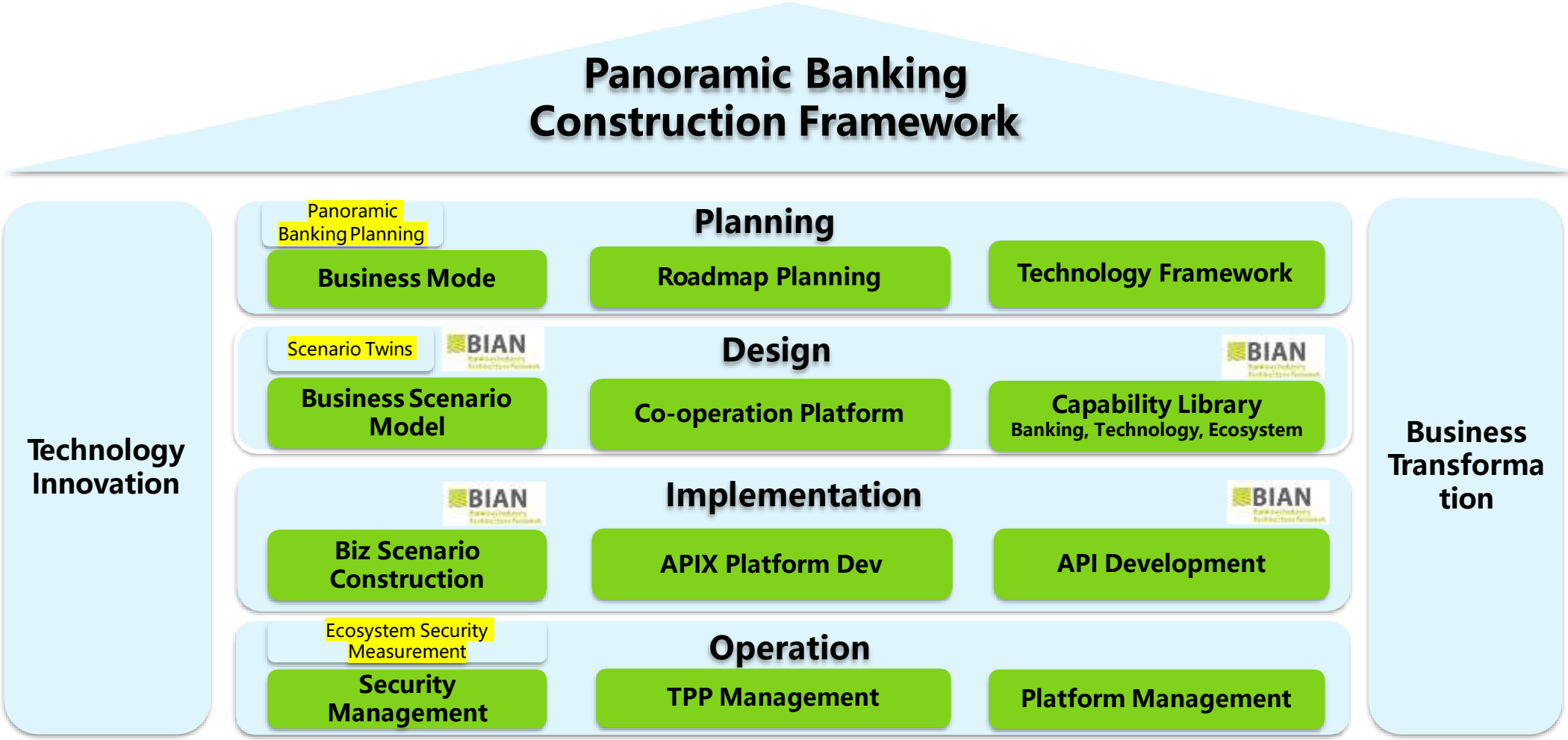
- **Realized value: Customers achieve their financial goals**
- Build open banking capabilities, integrate fintech products, and provide customers with better services. With the progress of technology, the distribution will be transformed from "data-driven" into "intelligent participation"

Industry Co-construction and Co-prosperity



- **The proposition of the Open Financial Union**
  - Pan-financial mutual assistance platform
  - Gather, create and share good resources
  - Create value for users and related parties
- **The main services of the Open Financial Union**
  - **Technology sector:** Promote the sharing of open banking platforms such as API Echange and technical platforms
  - **Business sector:** Promote extensive cooperation among members in various fields of business operation
  - **Ecosystem sector:** Realize the sharing of ecosystem operation capability, enhance ecosystem vitality, and realize scale advantage

Panoramic Banking Construction Framework



"Adopting a set of completed and standardized methods to enhance panoramic banking core competitive capability."

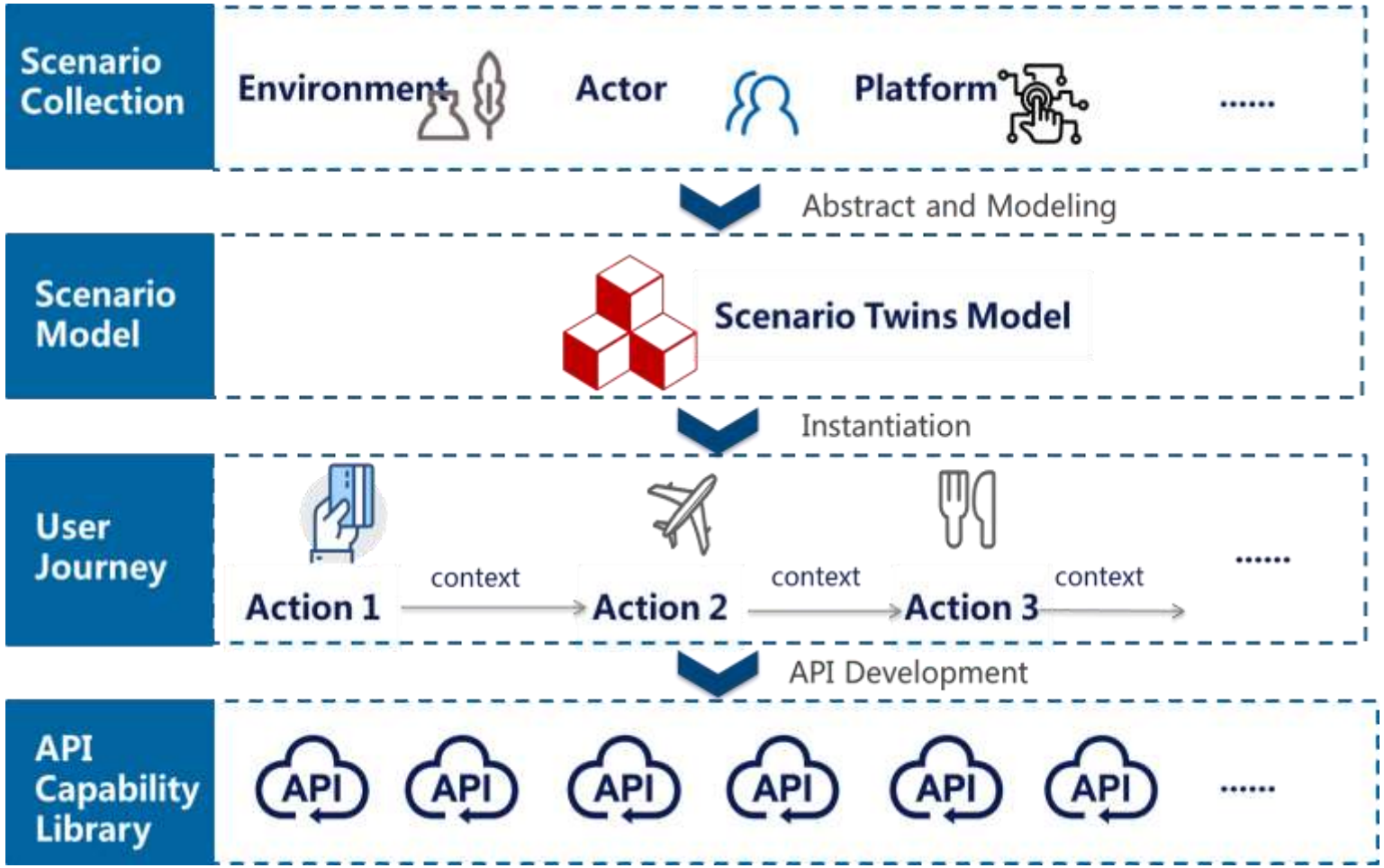
浦发银行 SPD BANK

Contents

浦发银行 SPD BANK

Scenario Twins

Scenario twins implementation includes scenario modeling, user journey design, open API design and capability construction.

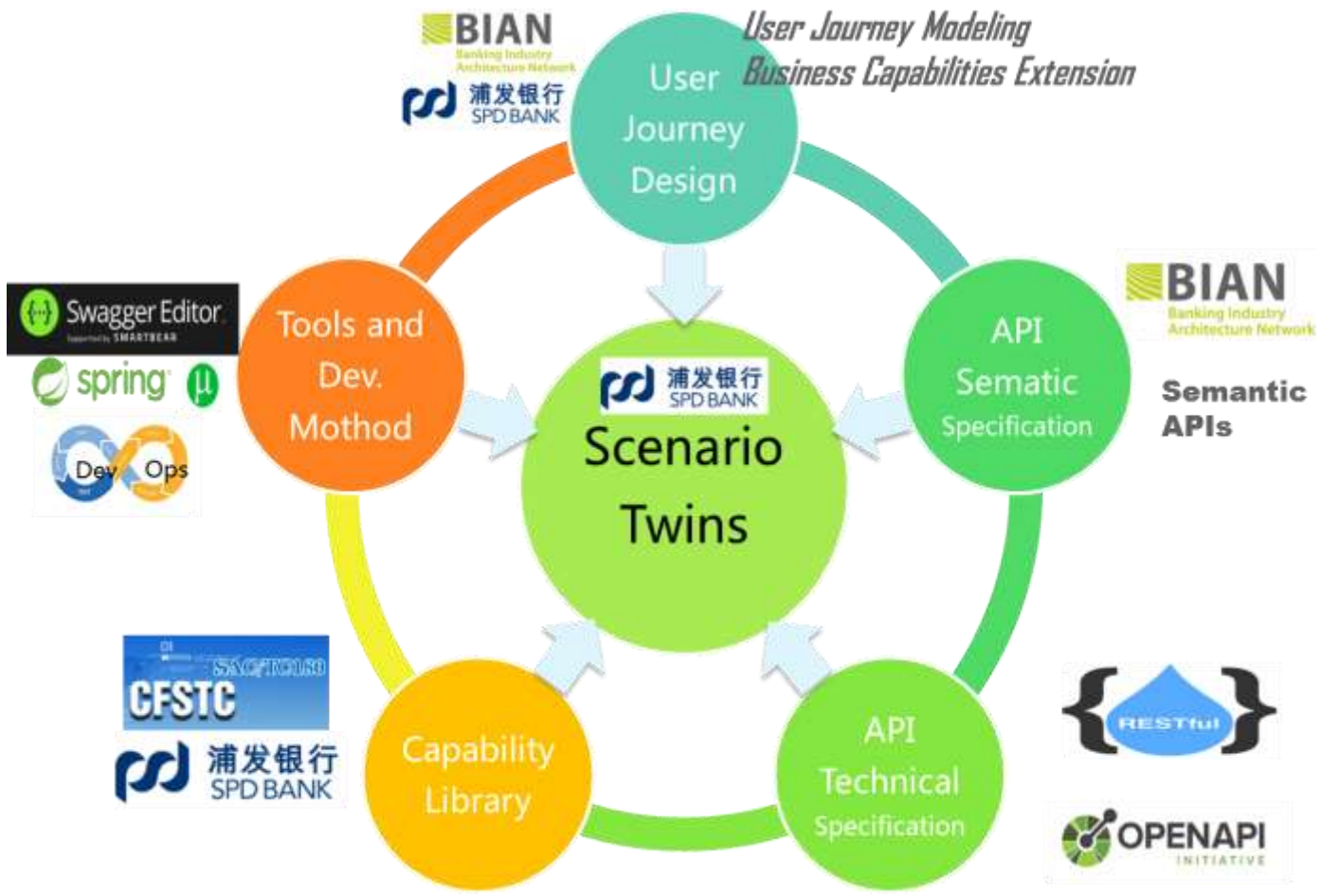


By structuring the scenario information such as actor, action, objective environment and service platform etc., the basic elements are standardized and the scenario is modeled

Design user journey template base on the model, constructs and normalizes the user journey data to twins the user activity in real world

Carry out semantic decomposition of user journey supporting services to make open API requirement, then develop APIs, forming the panoramic banking capability

Panoramic Banking Technic Combination



1 User Journey Design

Based on BIAN standard, introduces the concept of user journey design, expands the service domain and business capability model, describes the scenario twins.

2 API Semantic Specification

With the help of BIAN semantic API specification, adapts to Chinese local requirements and creates the API design specification.

3 API Technical Specification

Using OPENAPI standard and RESTful API best practices, creates API technical specification.

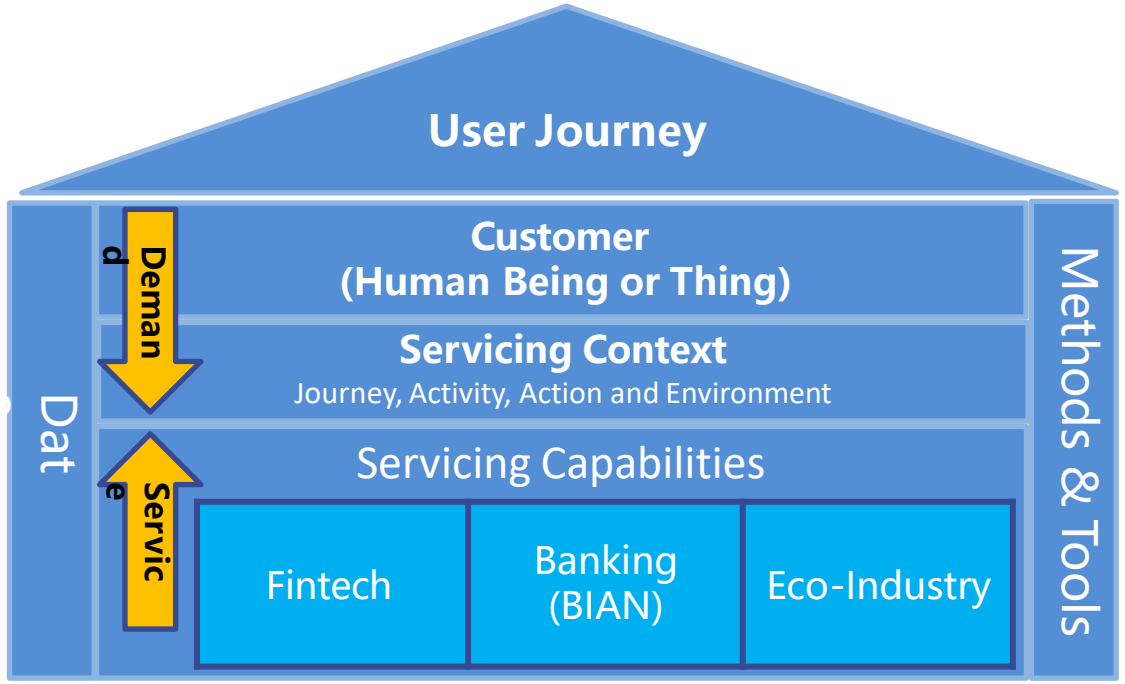
4 Capability Library

Be compatible with the *Commercial Bank Application Program Interface Security Management Specification* issued in 2020, builds panoramic banking capability directory.

5 Tools and Development Method

Swagger Editor tool is used to define API and generate microservice framework code. DevOps pipeline is used to standardize the process of design, development, testing and production.

User Journey Model



Service demand comes from customer's activities in ecosystem. The activities are not totally alone, some of them are closely related with each other even when they belong to different industries. To make the service more smart and pleasant to customer, the service provider should not care only about the activities themselves, but also the relationship and the environment of the activities

The information of the linked activities to accomplish a goal and their environmental information assembly is user journey.

With user journey information, the bank could get deep insight of customer, then could possibly serve the personalized, contextual demand.

User Journey Model is base on User Journey Map and is enhanced with completed information. It is the structured information mirroring the customer's behavior and its background in real world.

User Journey Map

Anna

"I want a Seamless Travel and shopping experience abroad"

As a sales of a big firm. Anna travels periodically. She manages work life balance by making the most pleasant experience each stop of the way. Travel should not be stressful experience.

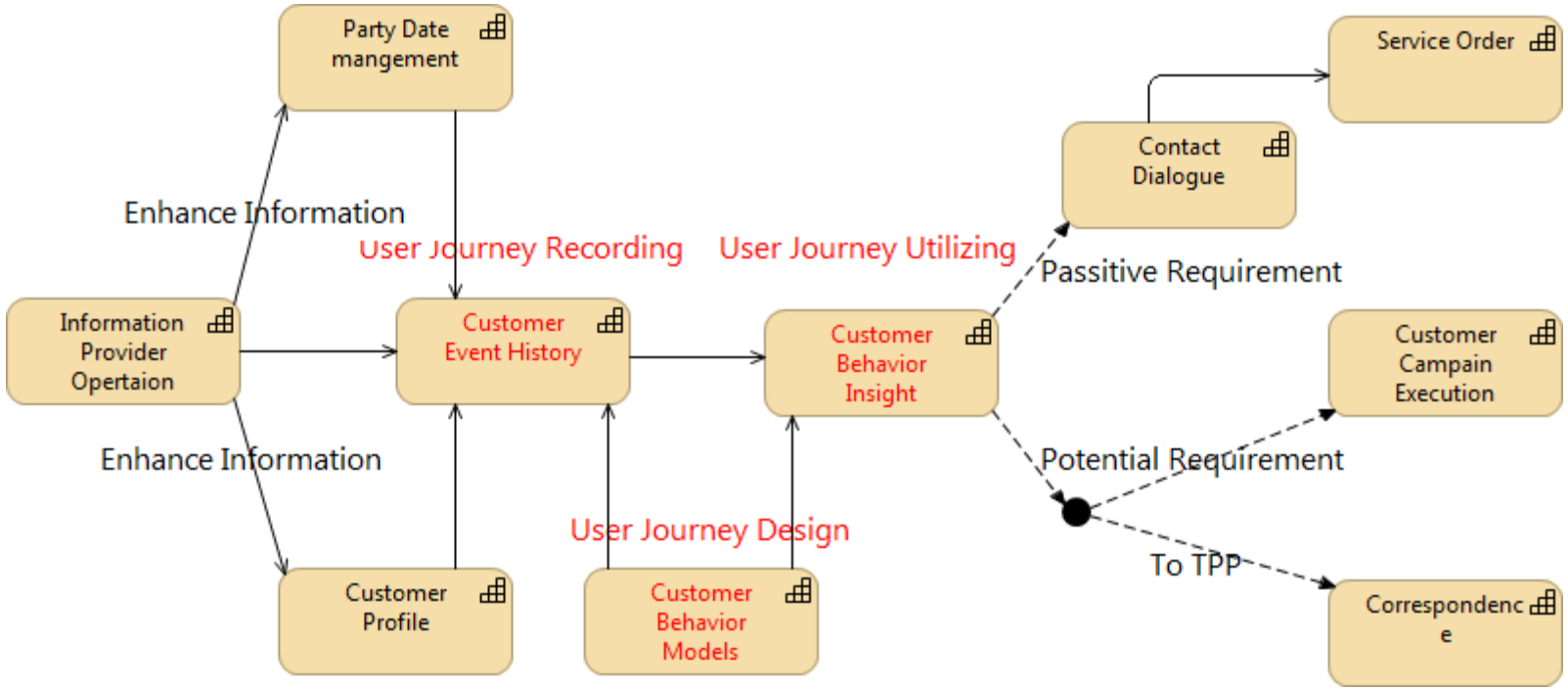
User Persona Expectation

Industry: **Travel** | **Bank** | **Retail**

User Journey: Travel Journey (context: Travel Itinerary, Budget, Social Event)

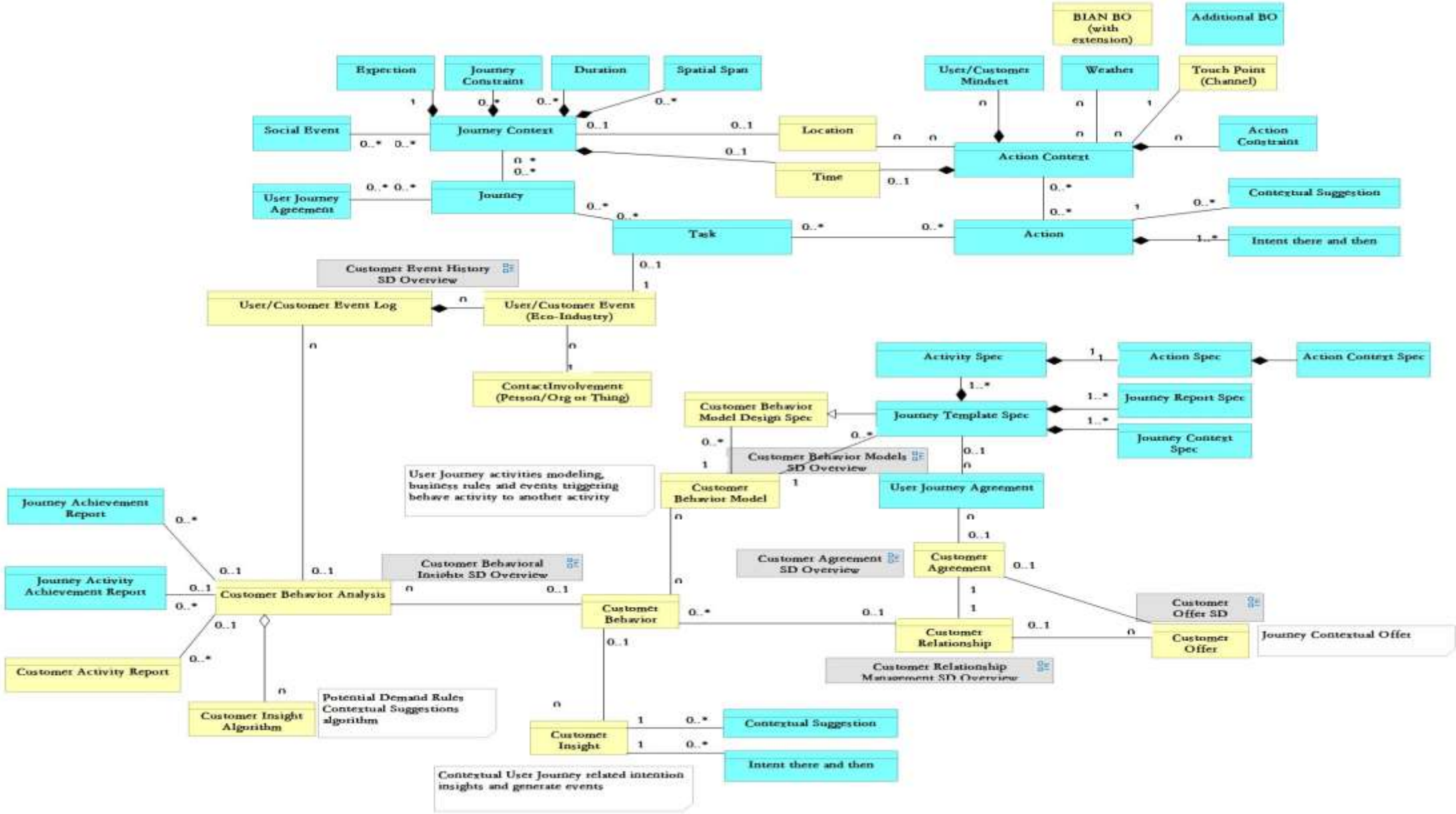
User Task	User Action	Action Context	Touch Point	User Achievement	User Experience
Book Travel	# 1		Travel agency app	H	Wow, only spend 15min finish all booking
Foreign Exchange	# 2		Travel agency app Or via bank api; Bank app	$	Wow, saved a lot of exchange fee
Travel	# 3		Merchant o2o	$	Wow, I got my most wanted bag in 50% discount in suggested mall without wasting shopping time.
Shopping	# 4		Bank app	Risk R	Wow, bank helped on mitigate potential FX risks
Travel	# 5		Travel agency app via tax api	H	Frustrated, shopping receipts should be paper-less and automated tax refund for me seamlessly
Reverse FX	# 6		Bank app	$	Wow, bank automatically convert excess foreign currency for me at optimal rate

BIAN SD Extention for User Journey



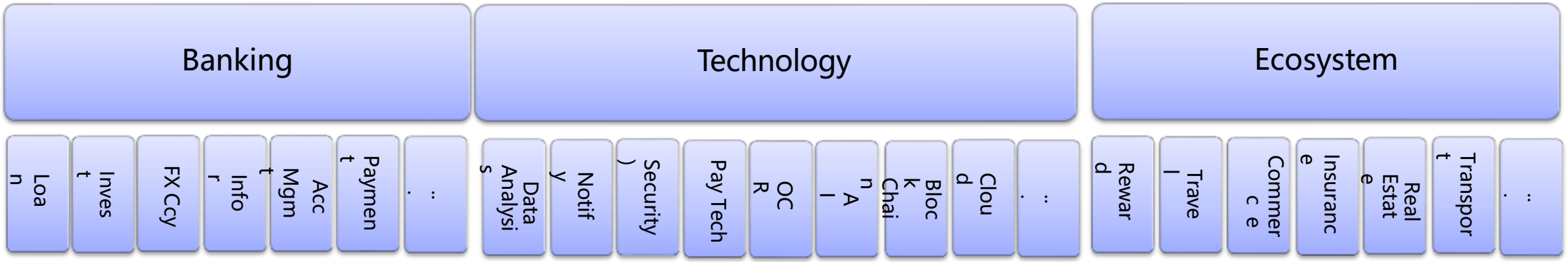
浦发银行 SPD BANK

BIAN BOM Extention for User Journey



浦发银行 SPD BANK

API Library



Banking: Traditional Banking product and Service → Classify align with current business line

Technology: Not relate to banking directly, base on digital technology, aims to improve performance → Focus on new technology, include OCR, Face Recognition, Block Chain, Big Data, Cloud etc.

Ecosystem: Not belongs to banking and finance serving ecosystem → Eyes on ecosystem, such as Transport, Real Estate, Food, E-commerce, Logistic etc.

The Red Hat Journey

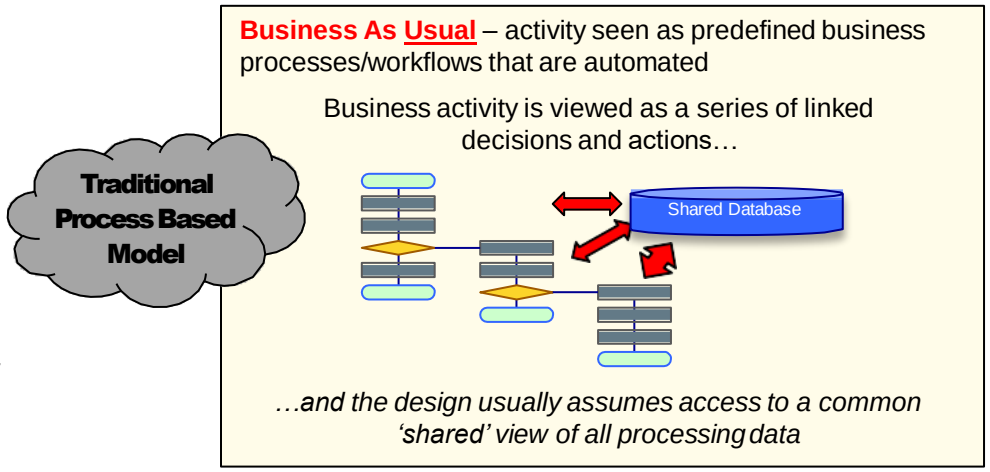
Examine considerations for the technical implementation of the BIAN standard needed for production applications.

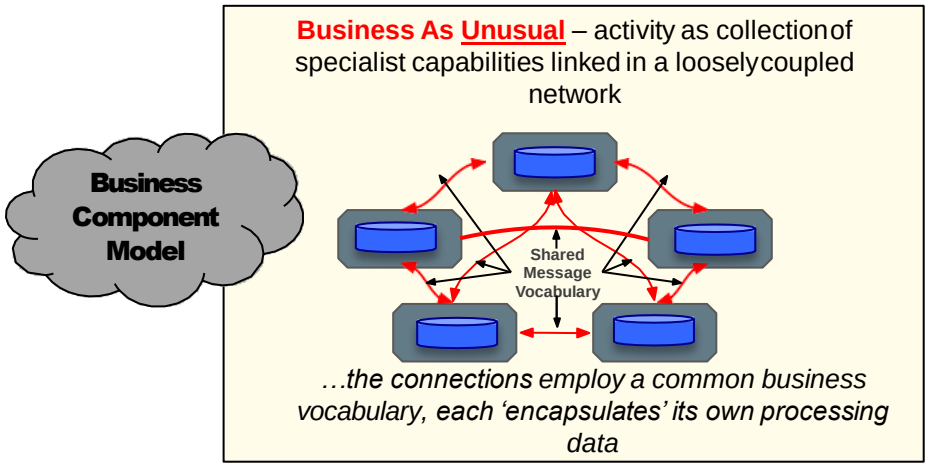
Q1: Can we interpret the BIAN model?	Q2: What is the ideal technical architecture?	Q3: What does a community need?

Answering Q1: Can we Interpret the BIAN Model?

BIAN in context of Red Hat enterprise open source

- Red Hat is used for:
  - mission critical applications
  - supporting services-based communications
  - highly distributed & automated production environments
- BIAN model design ensures:
  - partitions are generic (while varying in their detail of implementation)
  - ensuring standardized definition for distinctive roles for each functional module
- Conclusion: Red Hat's open modular platform can support the what is needed for BIAN-based application development and execution





Red Hat

Open specification ⇨ open source collaboration ⇨ enterprise

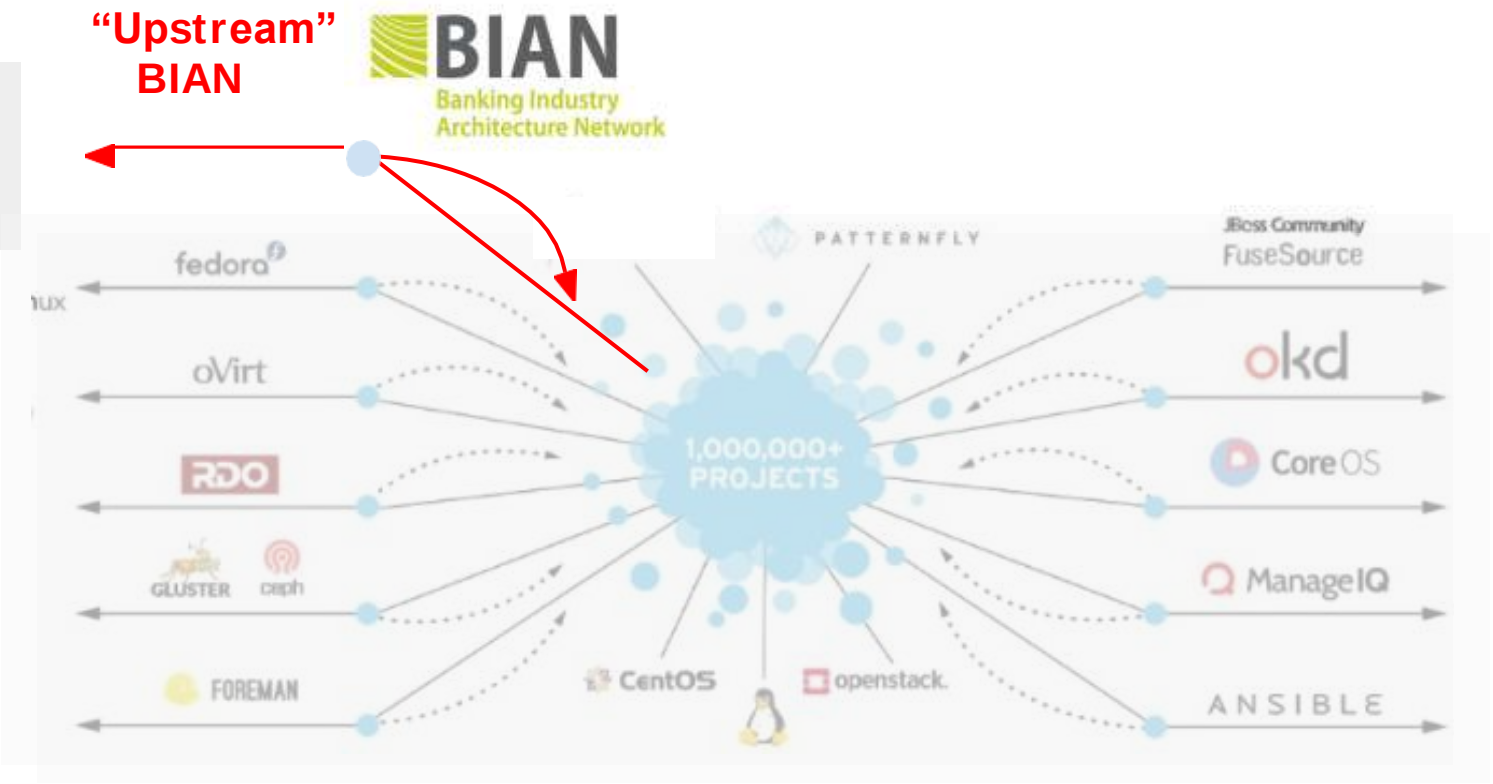
Visit https://www.redhat.com/en/about/open-source-program-office/contributions for details on Red Hat's community involvement, which spans span software foundations, operating systems, containers, application services and identity, security, developer tools, storage and compute, desktop, and operations

Red Hat

Industrialized financial services business modules & ecosystem

FinServ marketplace

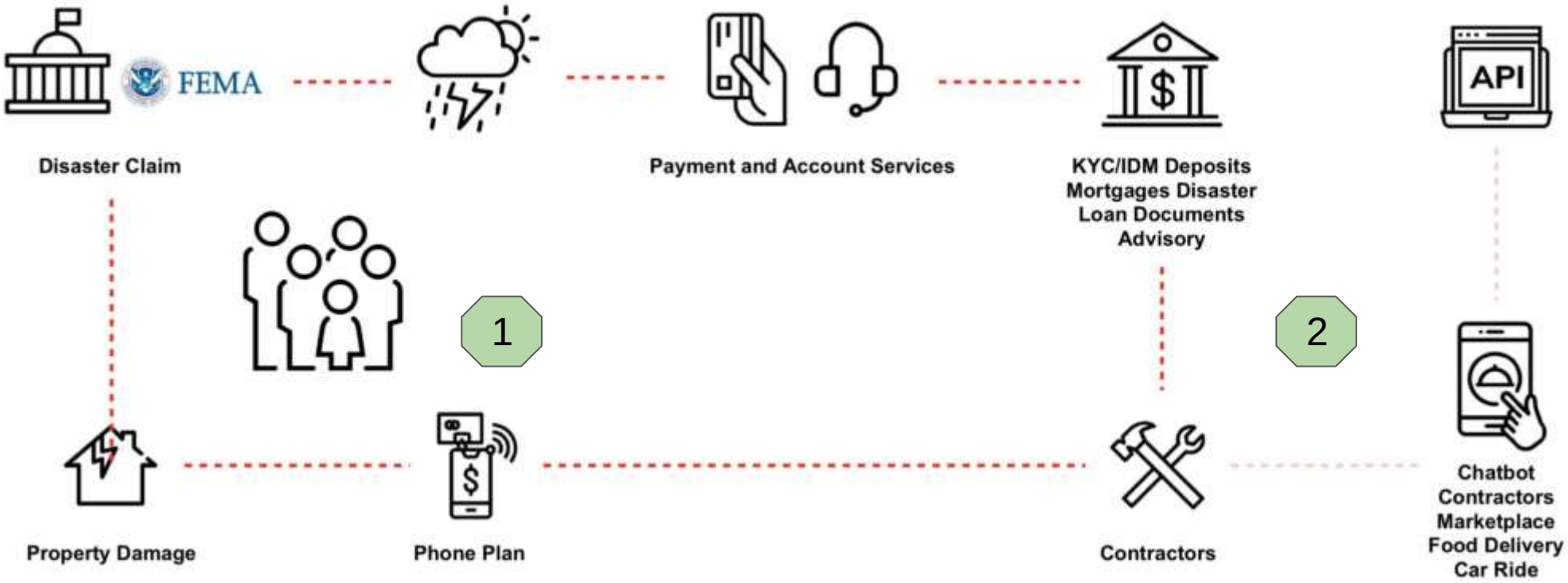
“Upstream” BIAN



Visit https://www.redhat.com/en/about/open-source-program-office/contributions for details on Red Hat’s community involvement, which spans span software foundations, operating systems, containers, application services and identity, security, developer tools, storage and compute, desktop, and operations

What is the role of the bank and providers in response to natural disaster

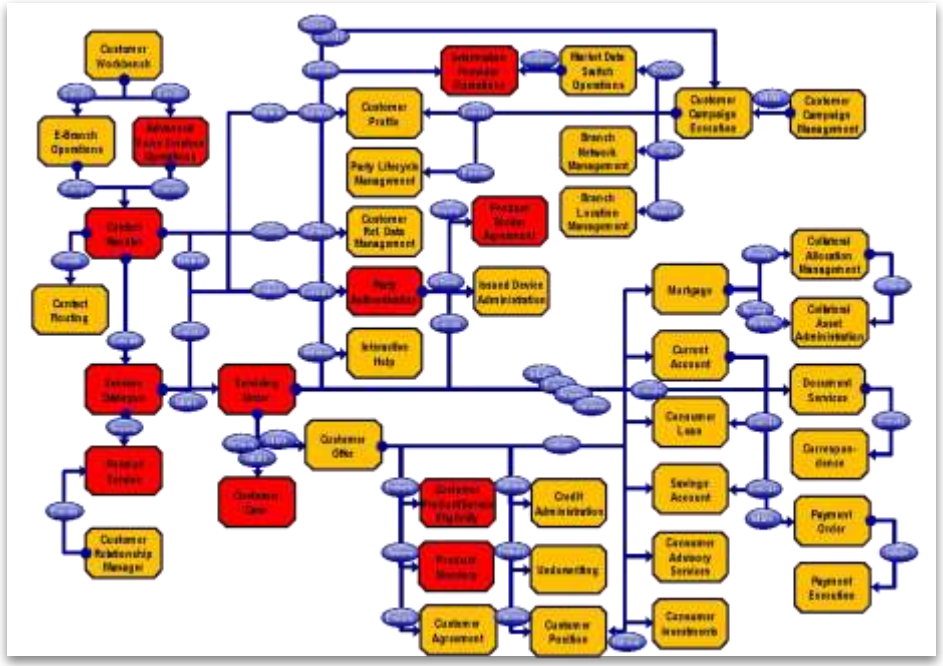
A BIAN perspective



Disaster Readiness | Immediate Assistance | Re-establish Access | Short Term Actions | Recovery Arrangement | Back to a New Normal

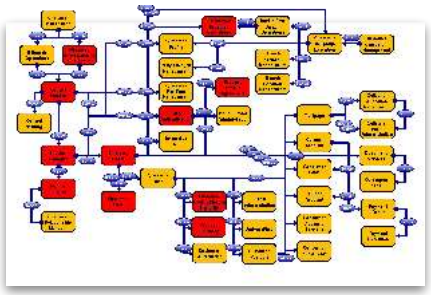
Red Hat

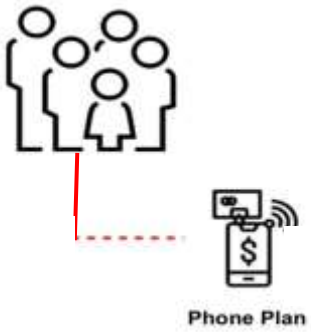
Modeling the story with BIAN service domains



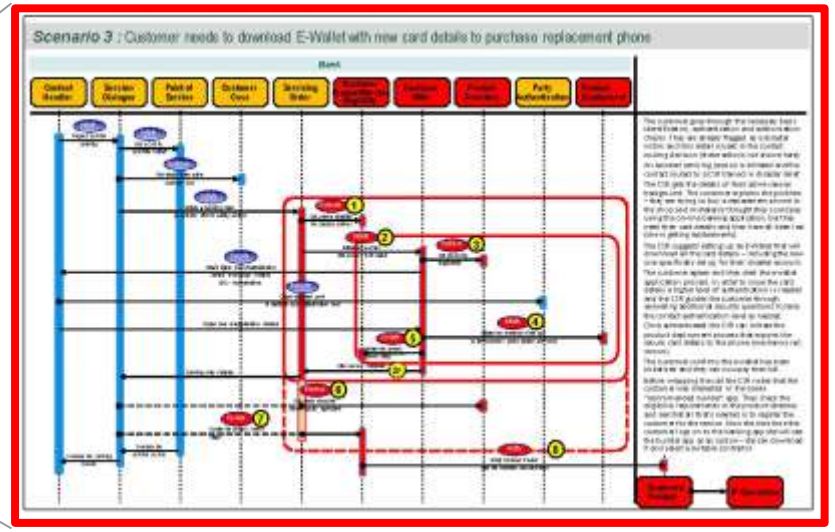
Red Hat

Modeling the story with BIAN service domains ..



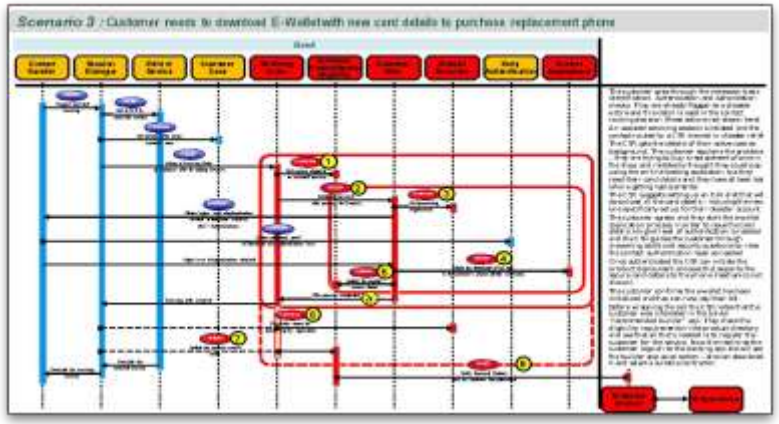


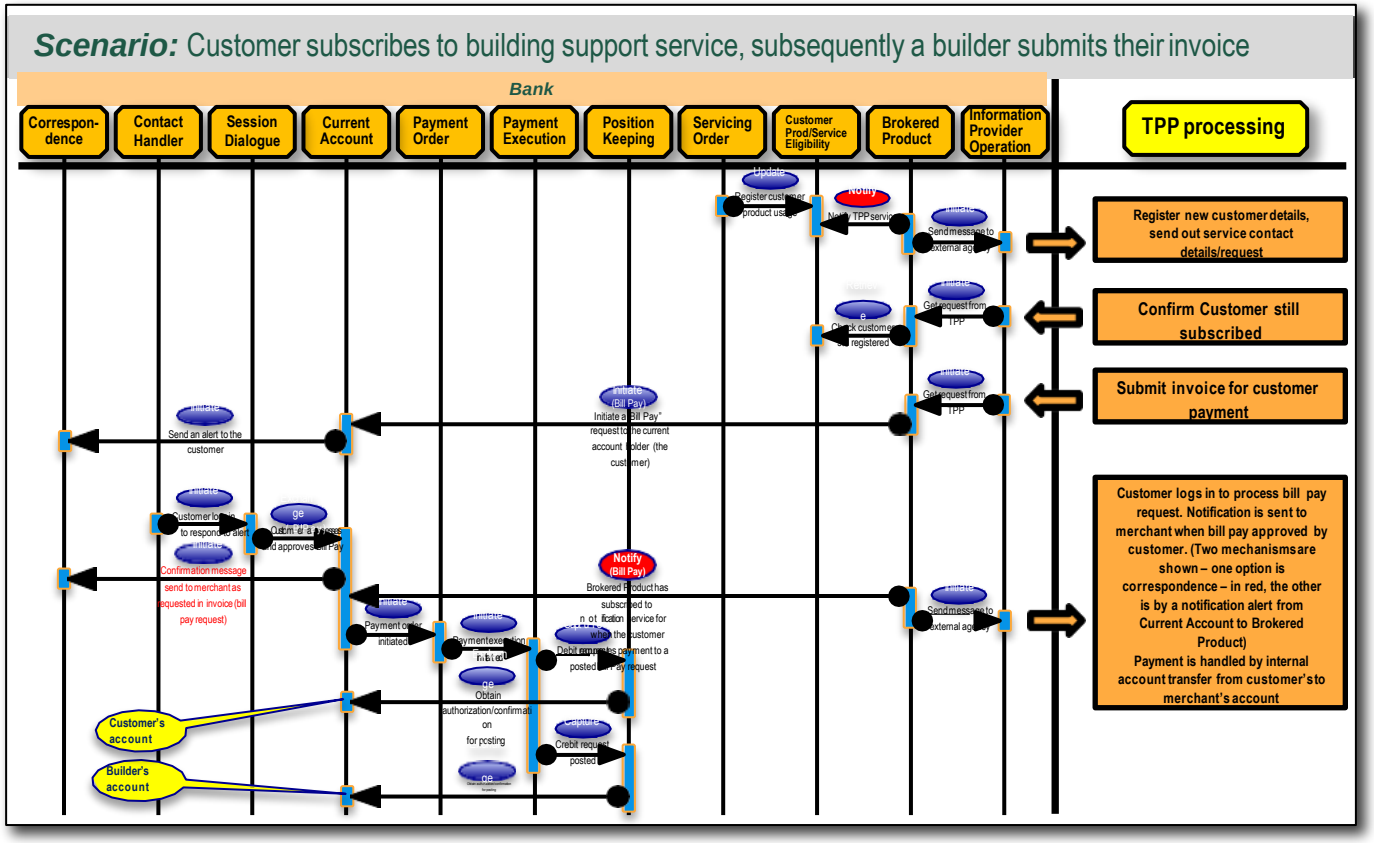
Re-establish Access



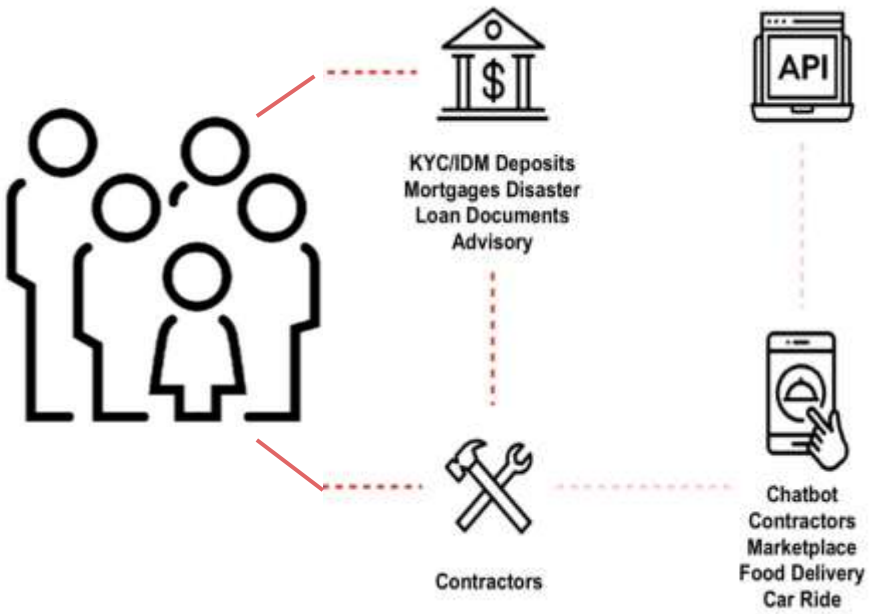
Red Hat

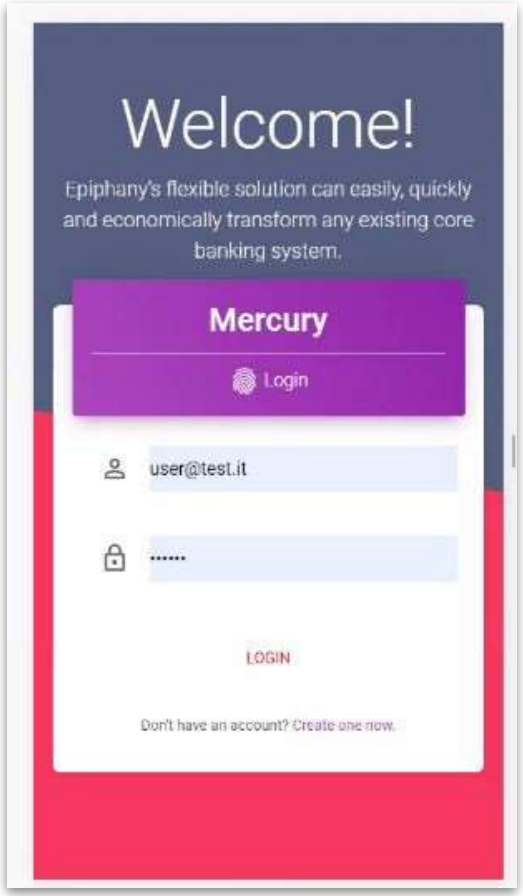
Extending the development team with partners

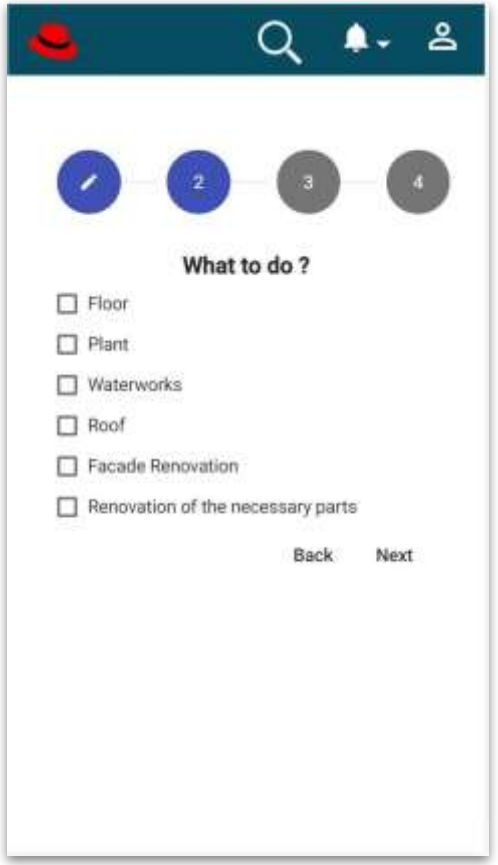




Building a working demonstration, collaboratively



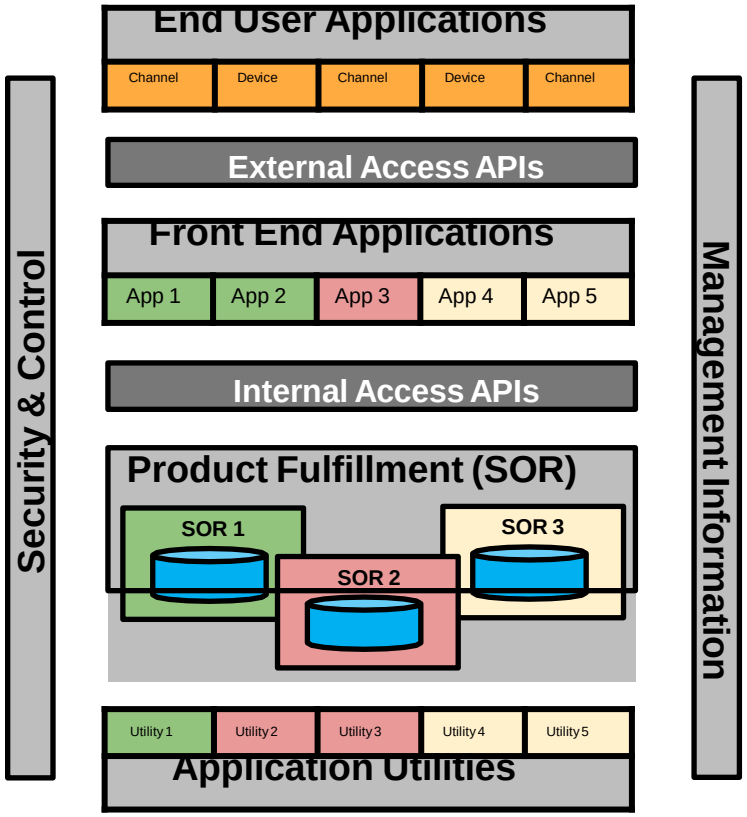


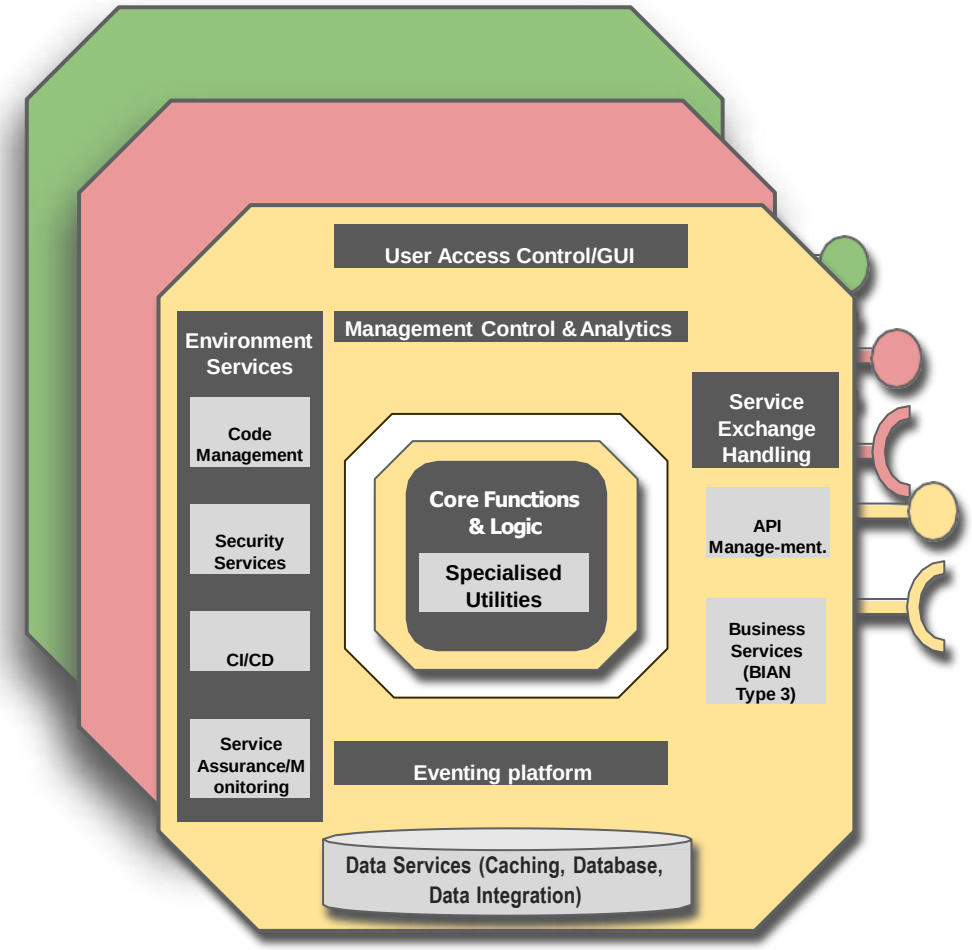


Red Hat

Answering Q2: What is the ideal technical architecture?

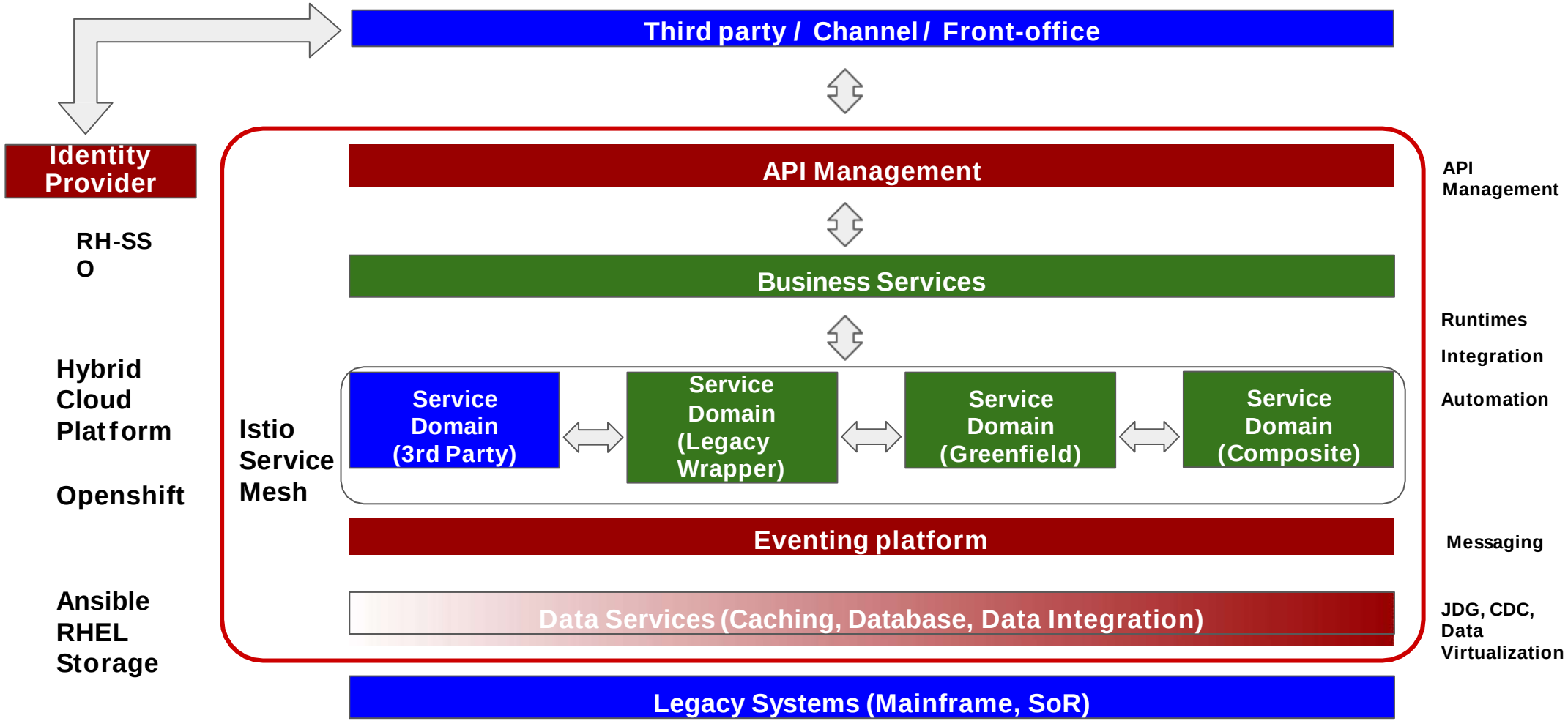
From a monolithic to a component architecture





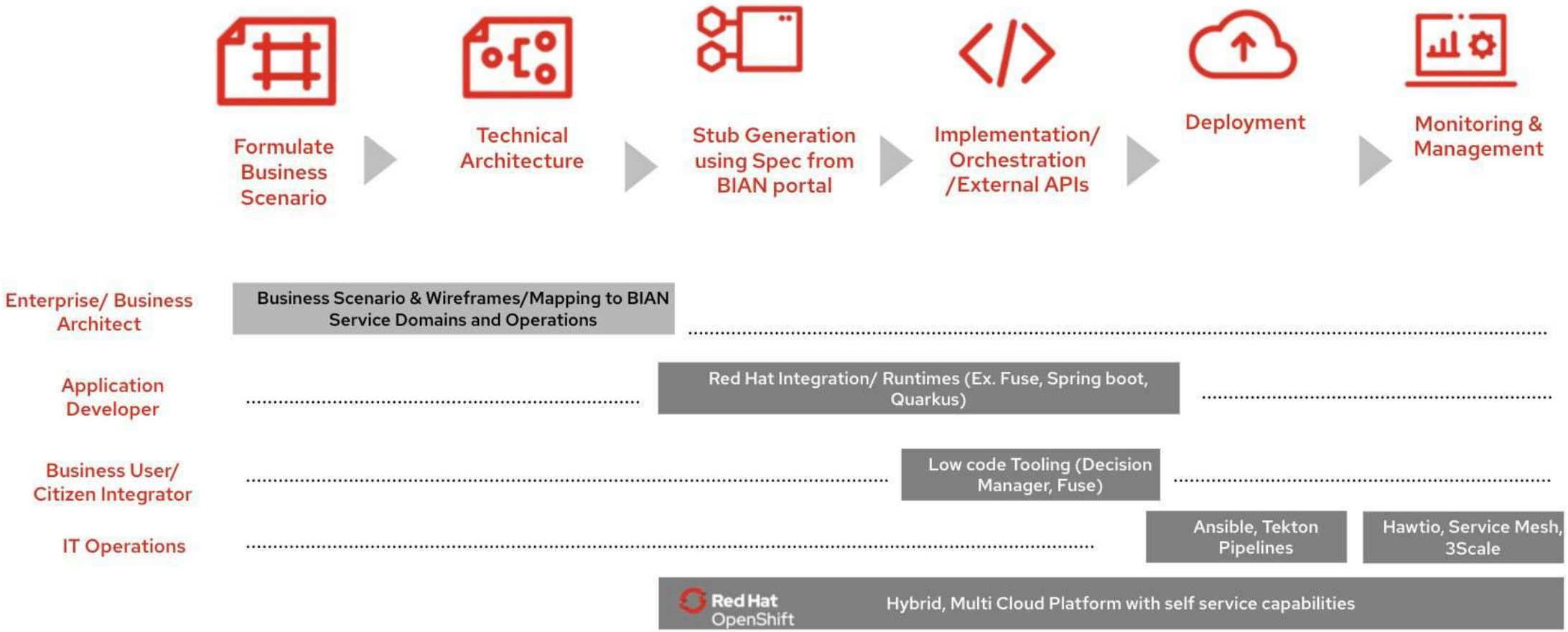
Red Hat

Red Hat perspective



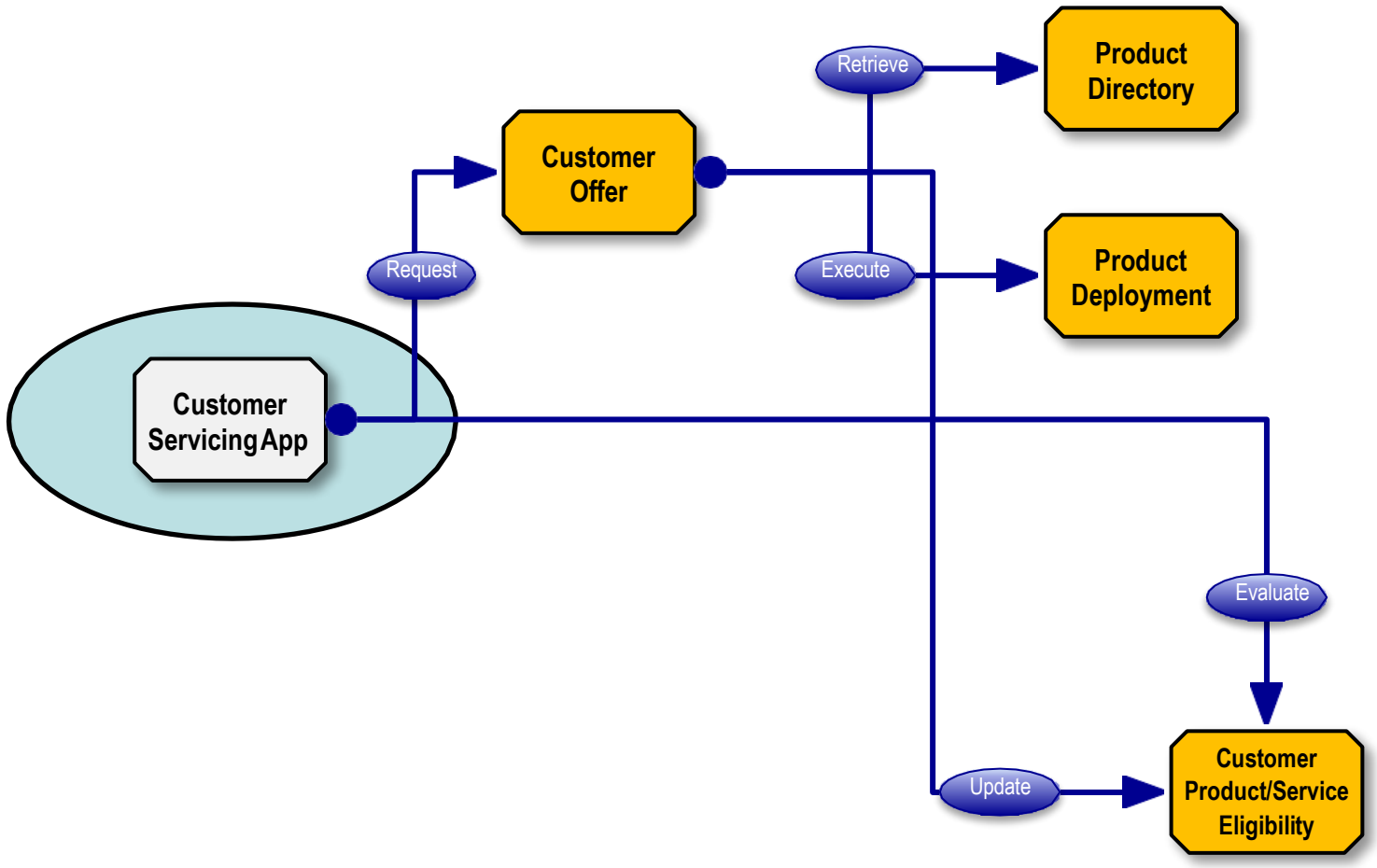
Red Hat

Our journey with the BIAN framework

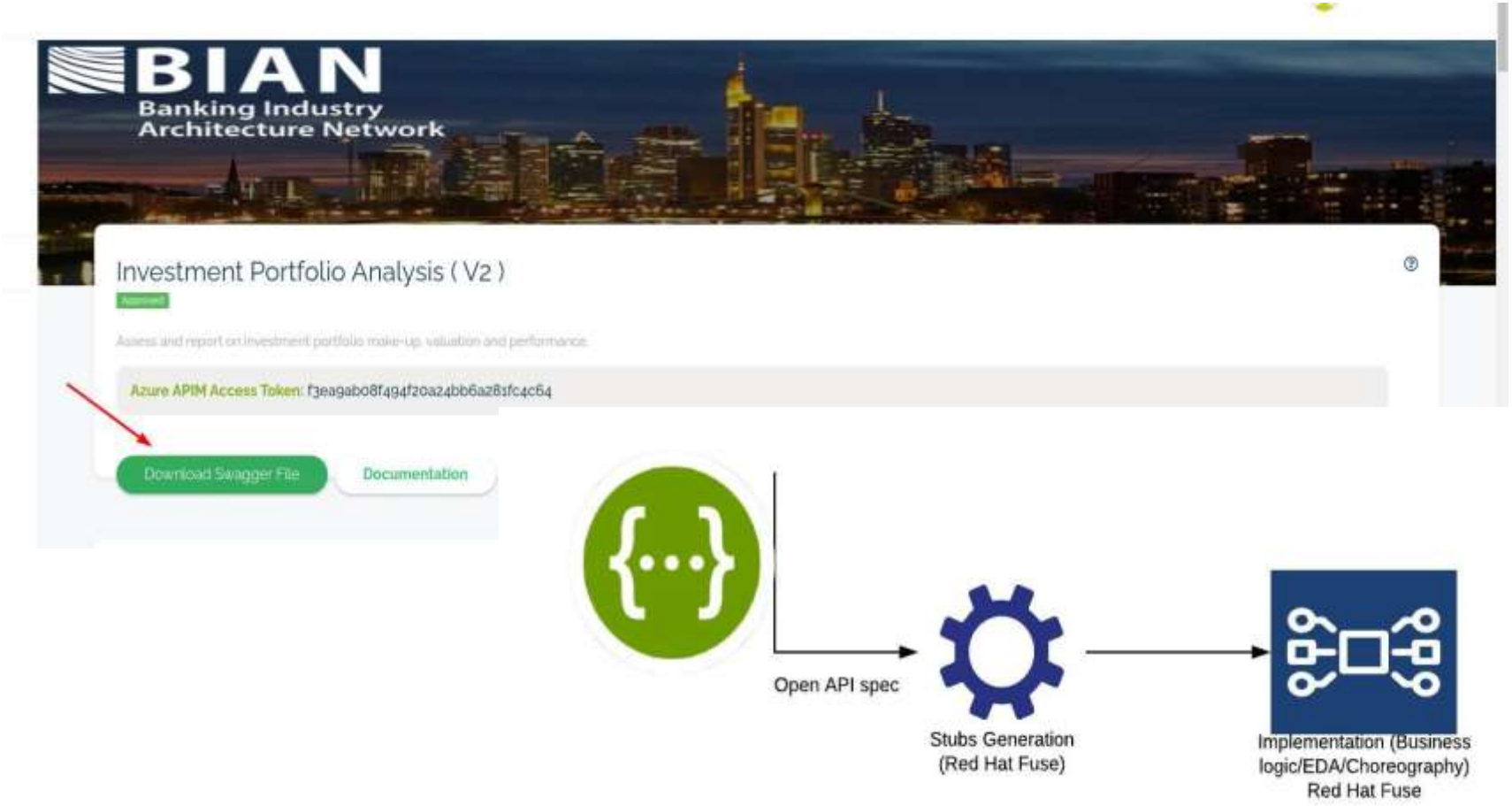


Red Hat

Minimal Viable Product (MVP) wireframe



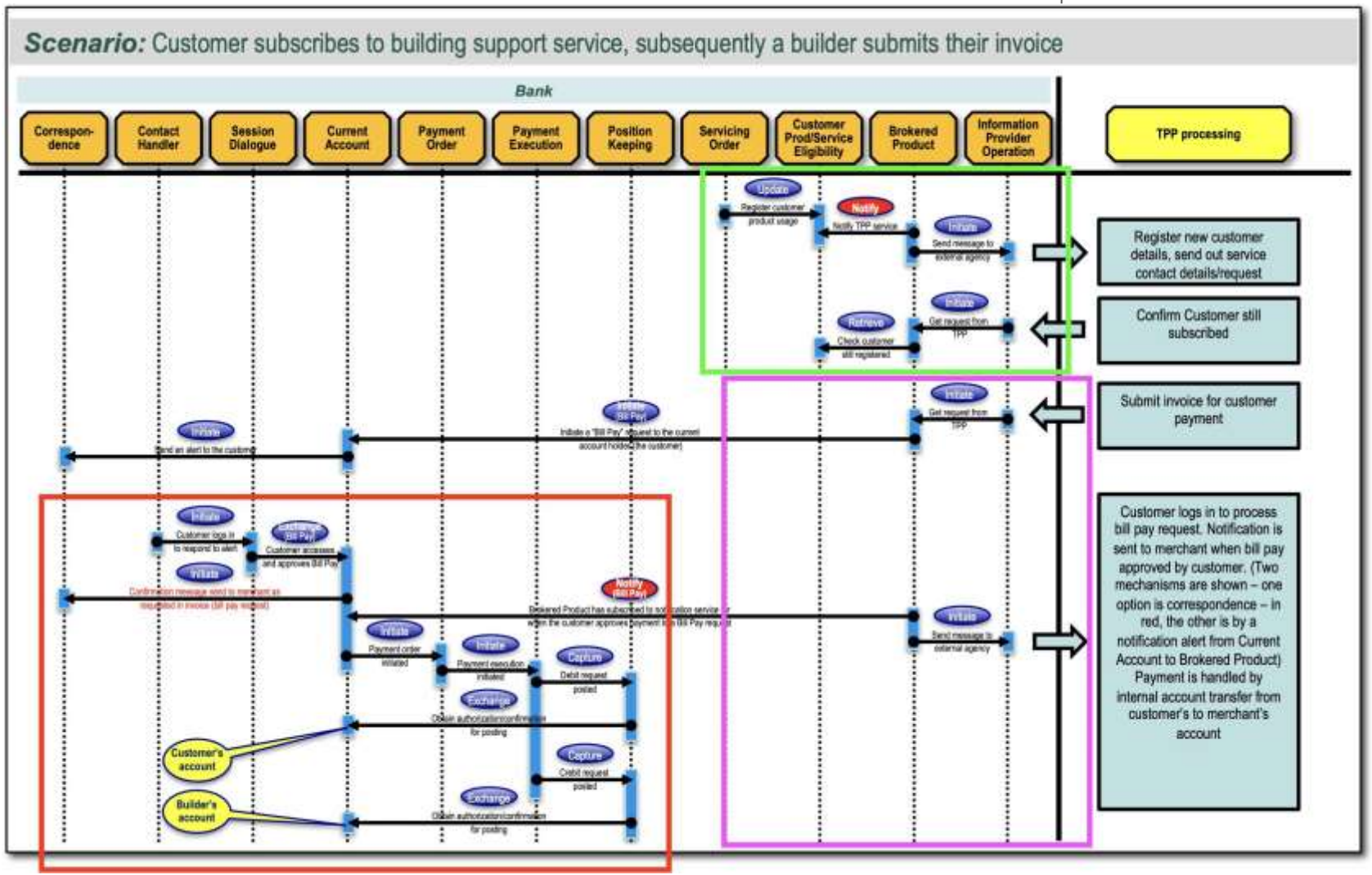
API generation using the BIAN specification



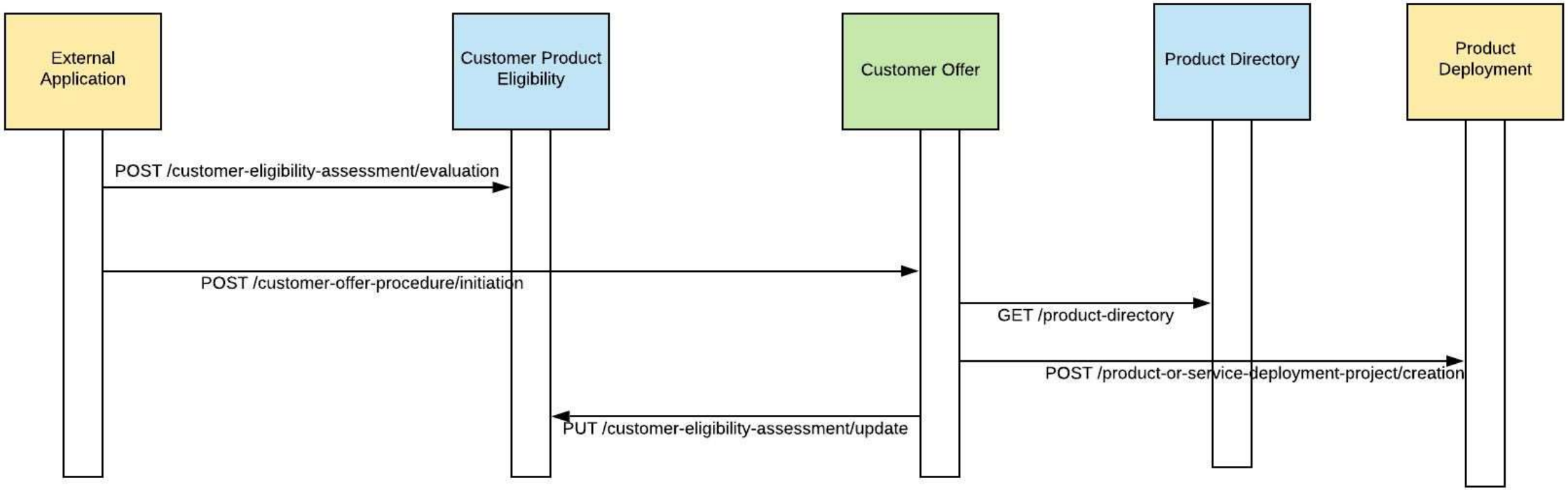
Red Hat

MVP Business Scenario

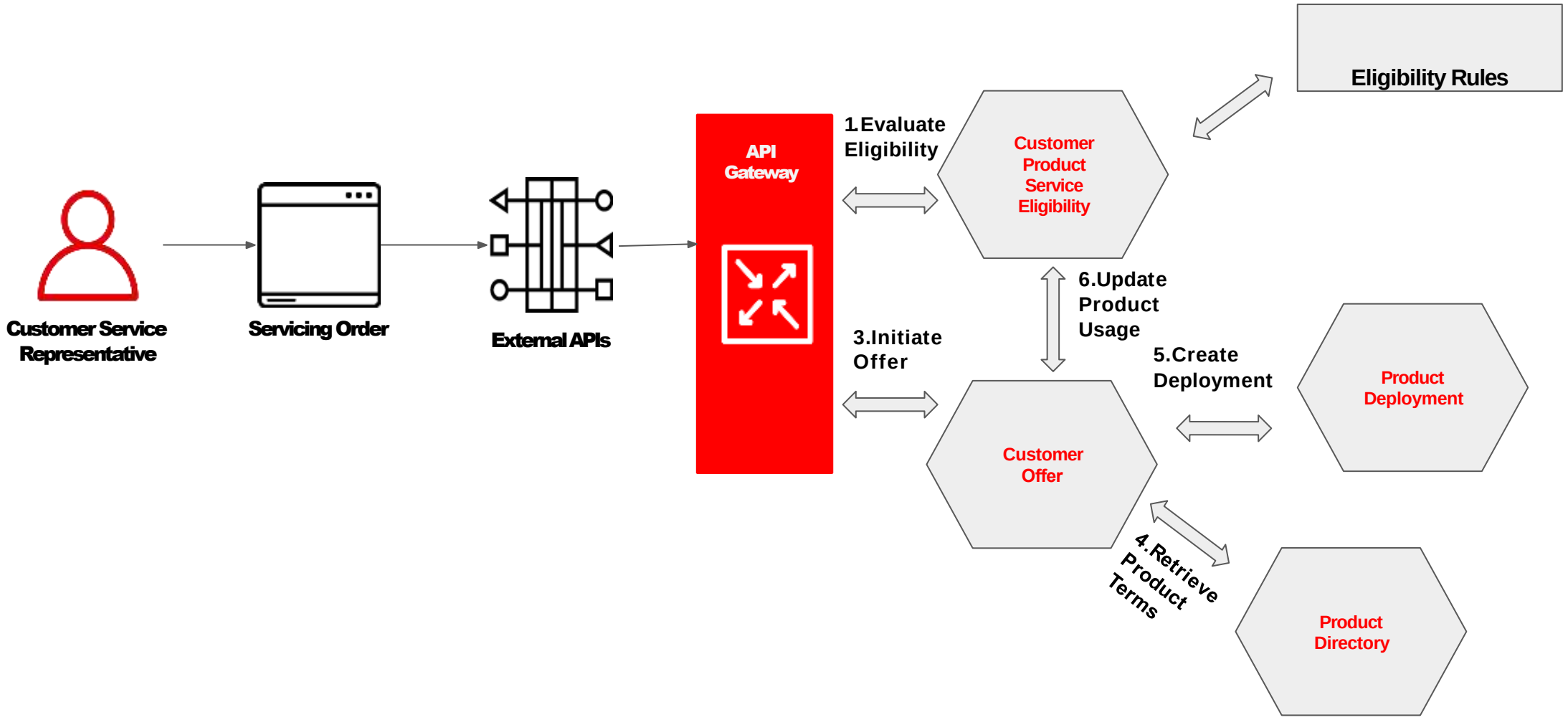
Bank | 3rd Party



From business scenario to design sequence

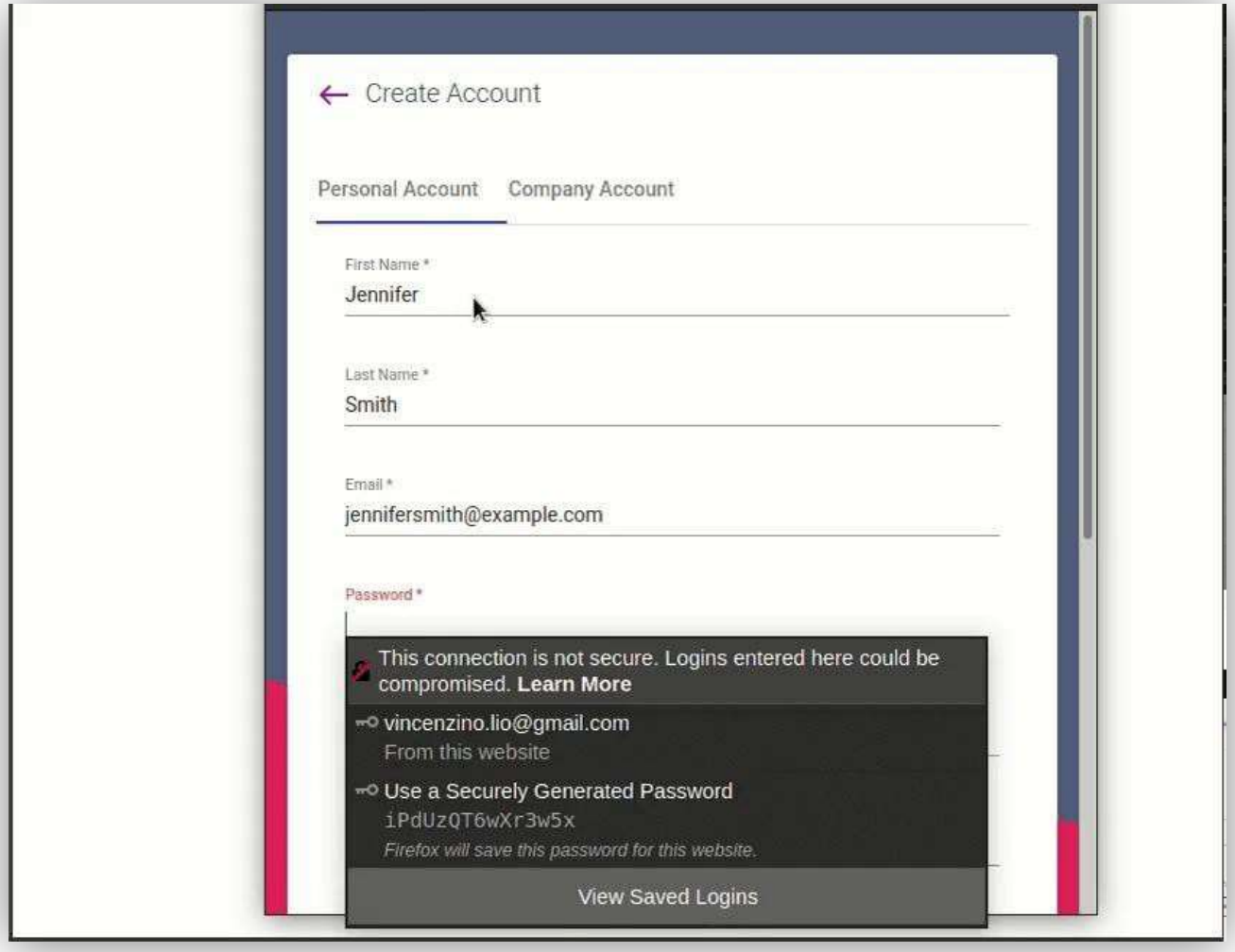


From design sequence to technical architecture



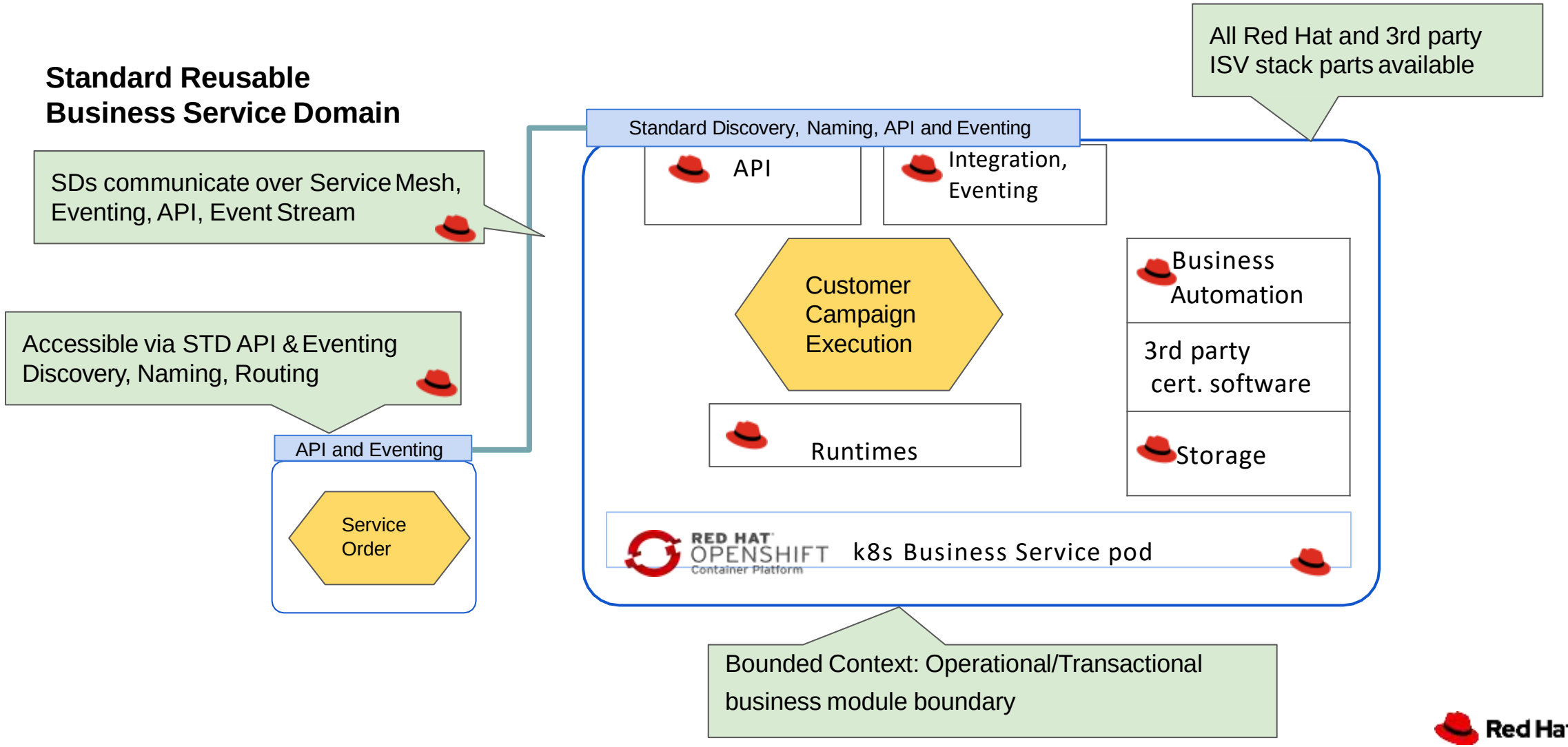
Red Hat

Demo in action



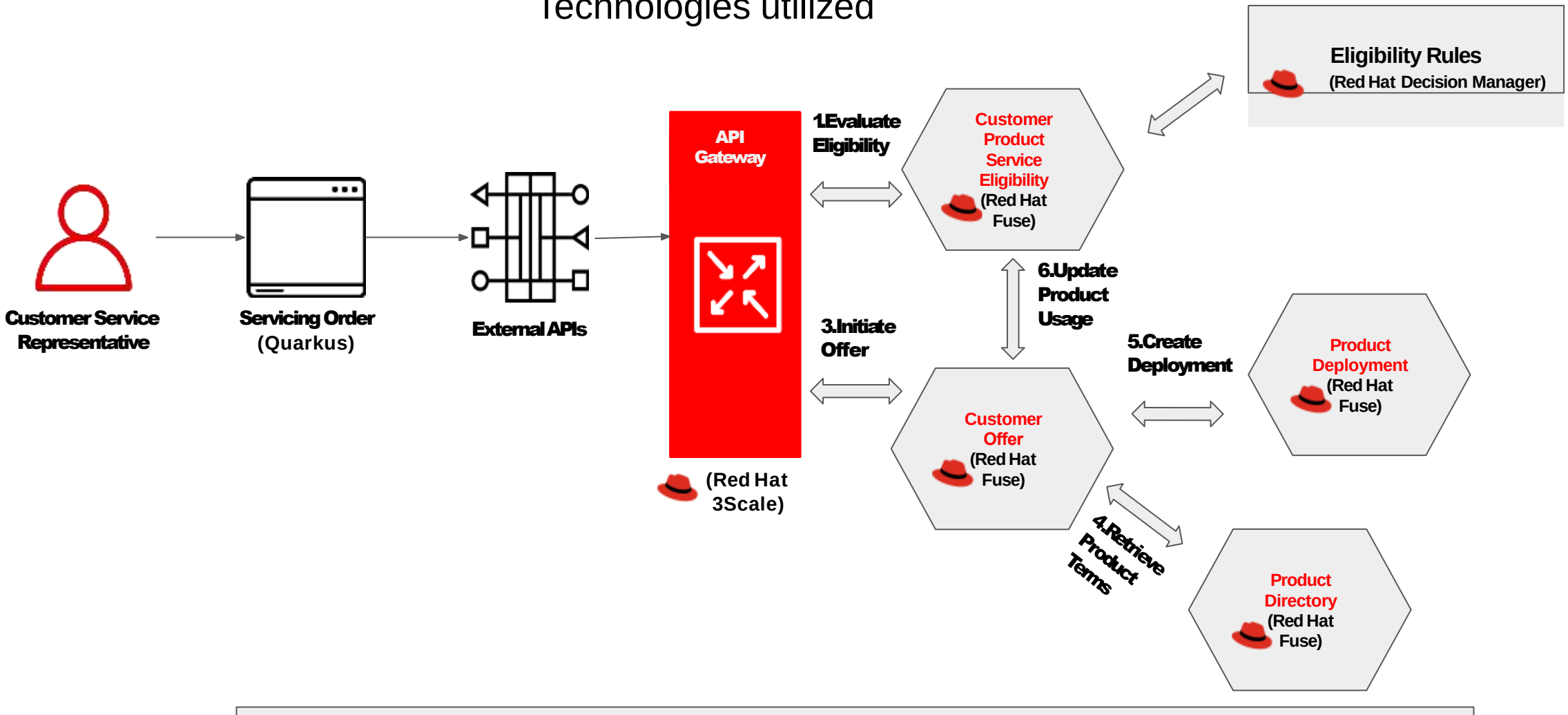
Red Hat

Anatomy of a business service domain



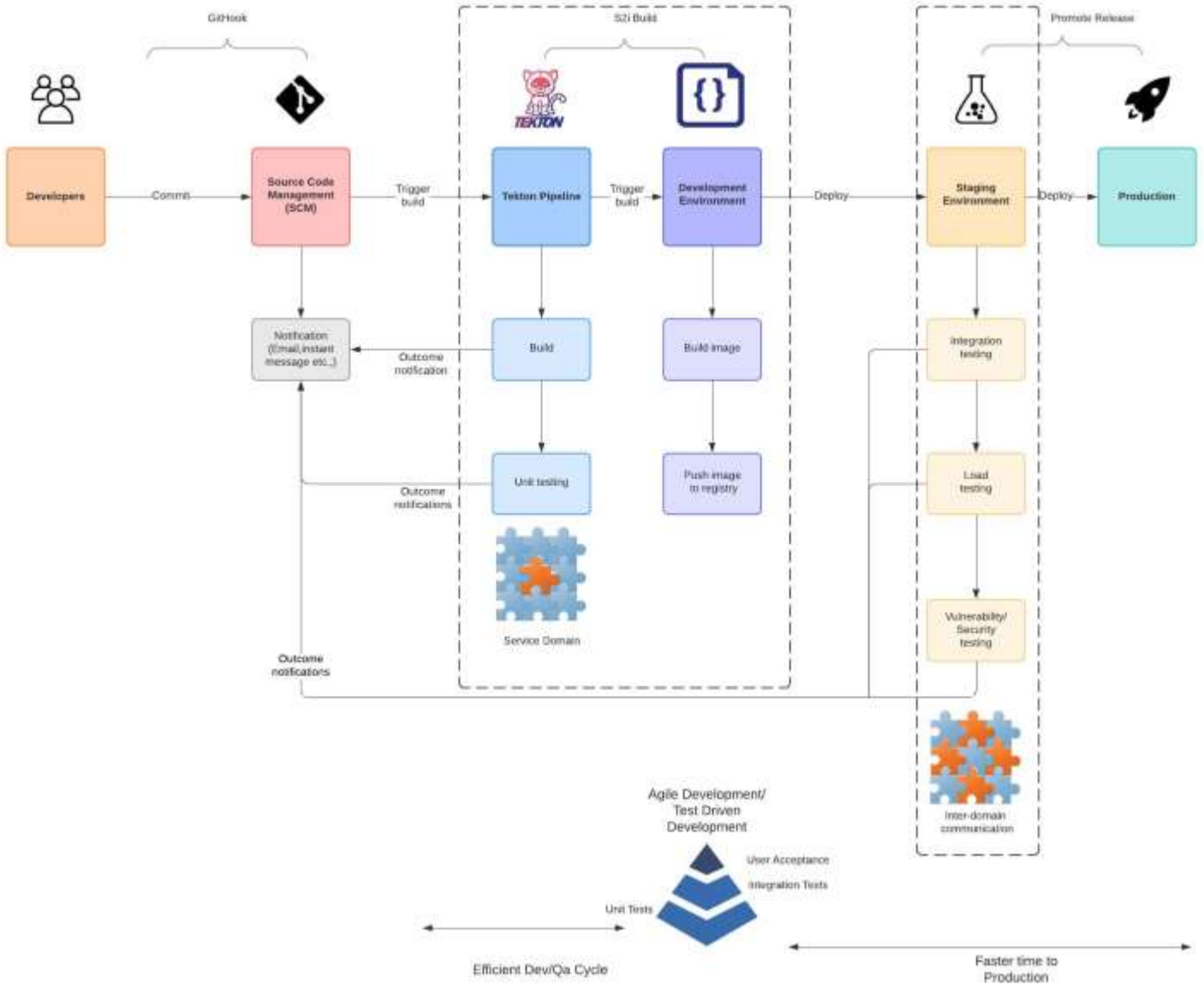
Red Hat

Technologies utilized



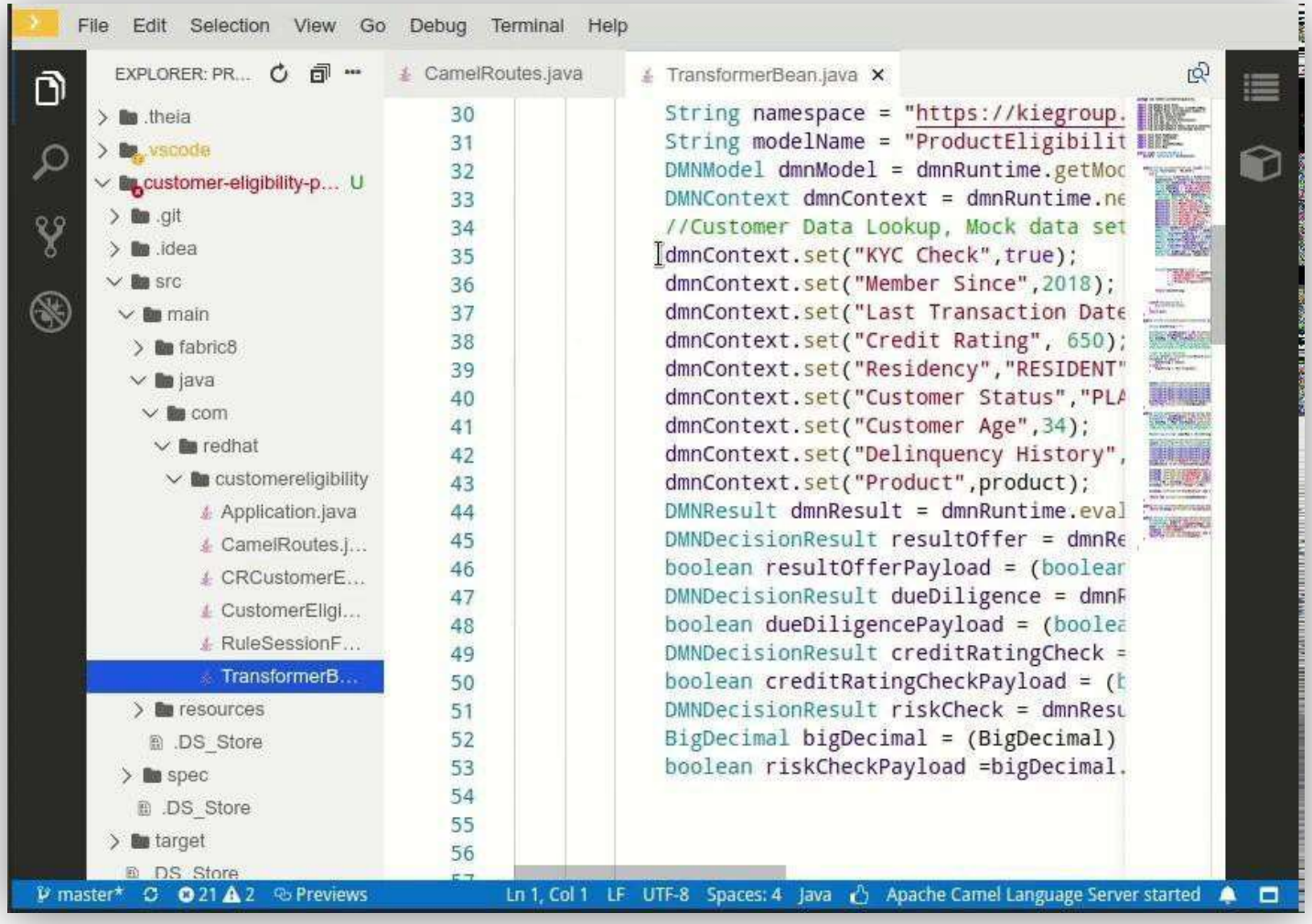
Red Hat

Production ready with Continuous Integration/Continuous Delivery (CI/CD)



Red Hat

CI/CD in action



Red Hat

Answering Q3: What would a community need?

Shared Understanding

An MVP approach in selected service domains can jump start adoption

Repeatable steps:

1. Define specific business activity
2. Identify supporting service domains & service exchanges
3. Develop minimal internal function and establish relevant service connections
4. Automate CI/CD
5. Implement feedback look from deployment to incrementally expand functional content

Applying the BIAN Concepts and Exchange Types

Real time ✓

Request-Reply (Sync/Async) requiring A2A integration, APIs, and Messaging.

Notification on Update ✓

Event Sourcing implemented using Change Data Capture and Messaging/Streaming

Delayed Response

Process Management (Async) often involves Human Tasks supported by workflow management and intelligent decisioning.

Best-time ✓

Request-Reply (Sync/Async) requiring mostly Messaging (includes Streaming)

Instant/Concurrent

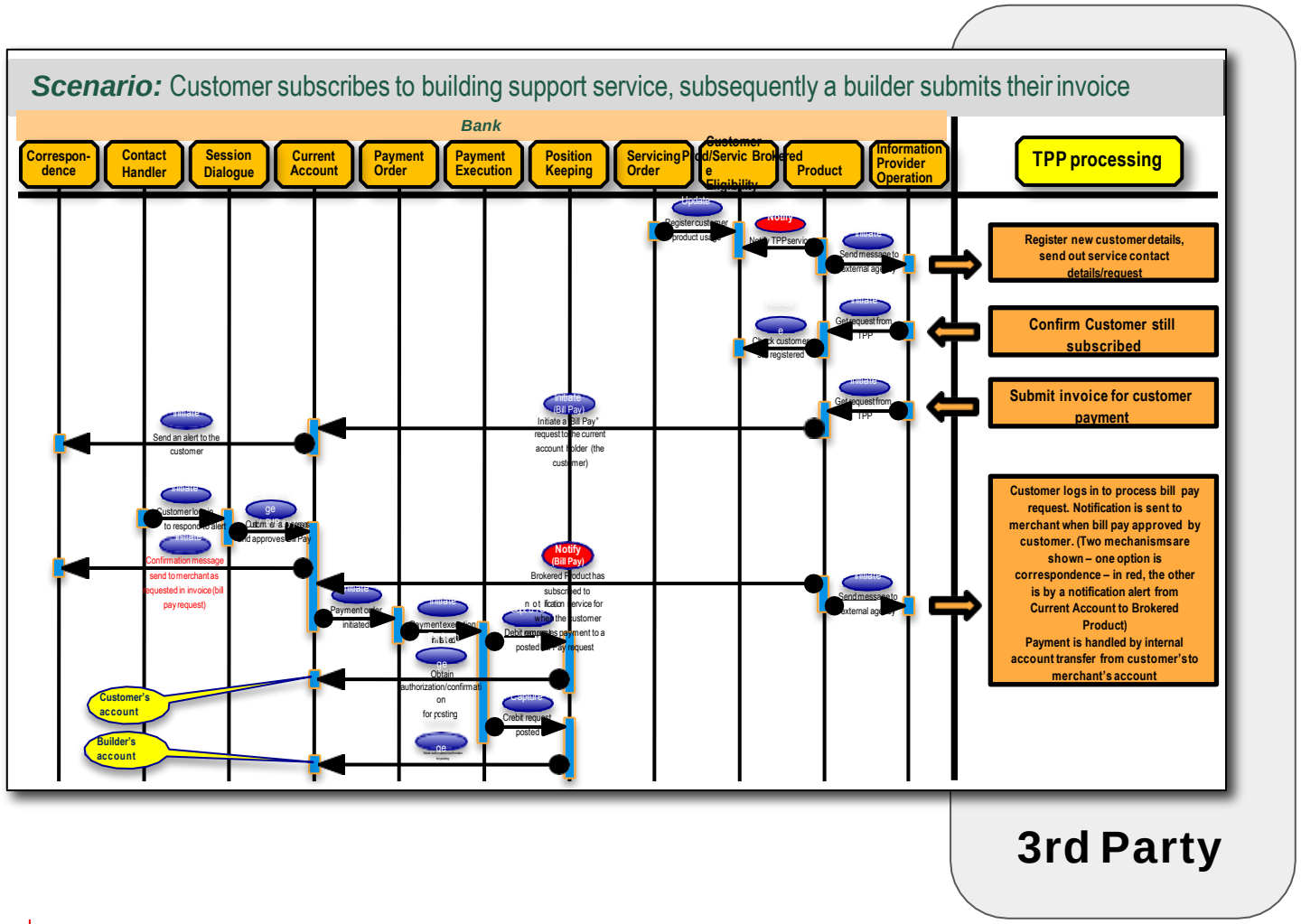
Shared Database for inter domain communication using Data Gateways / Services

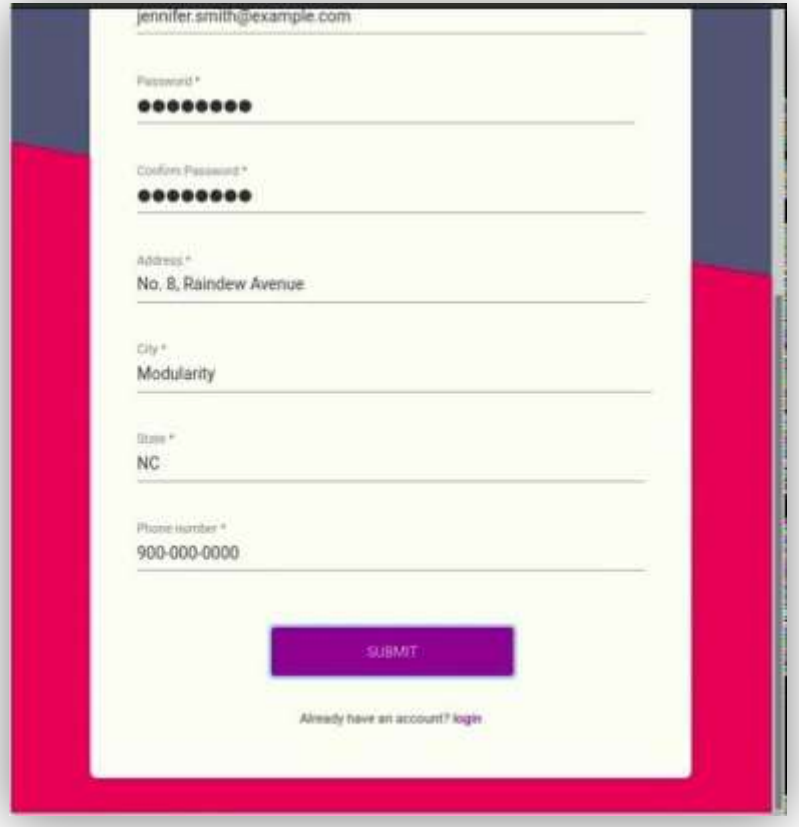
Scheduled Synchronization

Batch Processing and ETL usually supported by Data Integration tools.

epiphany

A collaborative environment





Red Hat

«epiphany

Epiphany ONE is a modular platform designed around an open collaboration concept: focused on enabling co-creation between banks and external partners to deliver new digital services

Red Hat

EPIPHANY ONE – An advanced open development environment leveraging the BIAN business component model…

BIAN Banking Industry Architecture Network ⟷ epiphany

3rd Party & Customer Integration
- External access flows
- FAPI/OIDC/Oauth2.0…

Epiphany ONE

- Covers the entire banking ecosystem:
  - banking transaction fulfillment;
  - customer facing applications; and,
  - third party (Fintech) integration.
- Leveraging BIAN to define standard business application 'components'.
- Cloud resident, container based, with support for rapid, incremental development and deployment

The foundation for a banking industry "App store"

Library of Core Banking System APIs/Adaptors

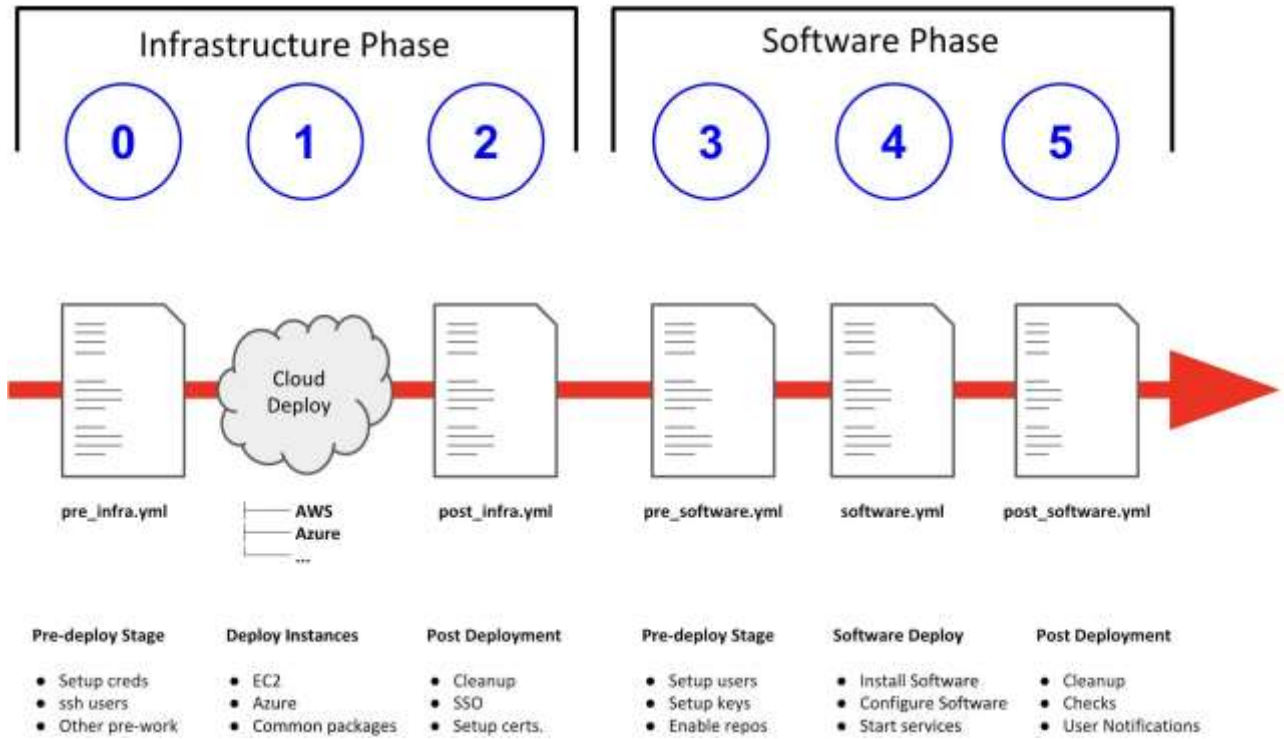
Development & Deployment
- Rich SKD, including…
- …Developer Sandbox
- Powerful Testing & Deployment Services

Red Hat

A place to practice and meetup

On-demand lab environment

- We established a platform supporting team development
- Automated provisioning available on-demand
- Easily extendable to a wider community
- Interested in lessons learned by other firms who have deployed



ANSIBLE

kubernetes

Red Hat

Santander

BIAN as a functional language - the journey (so far!) toward fully compliant Domain Driven landscape.

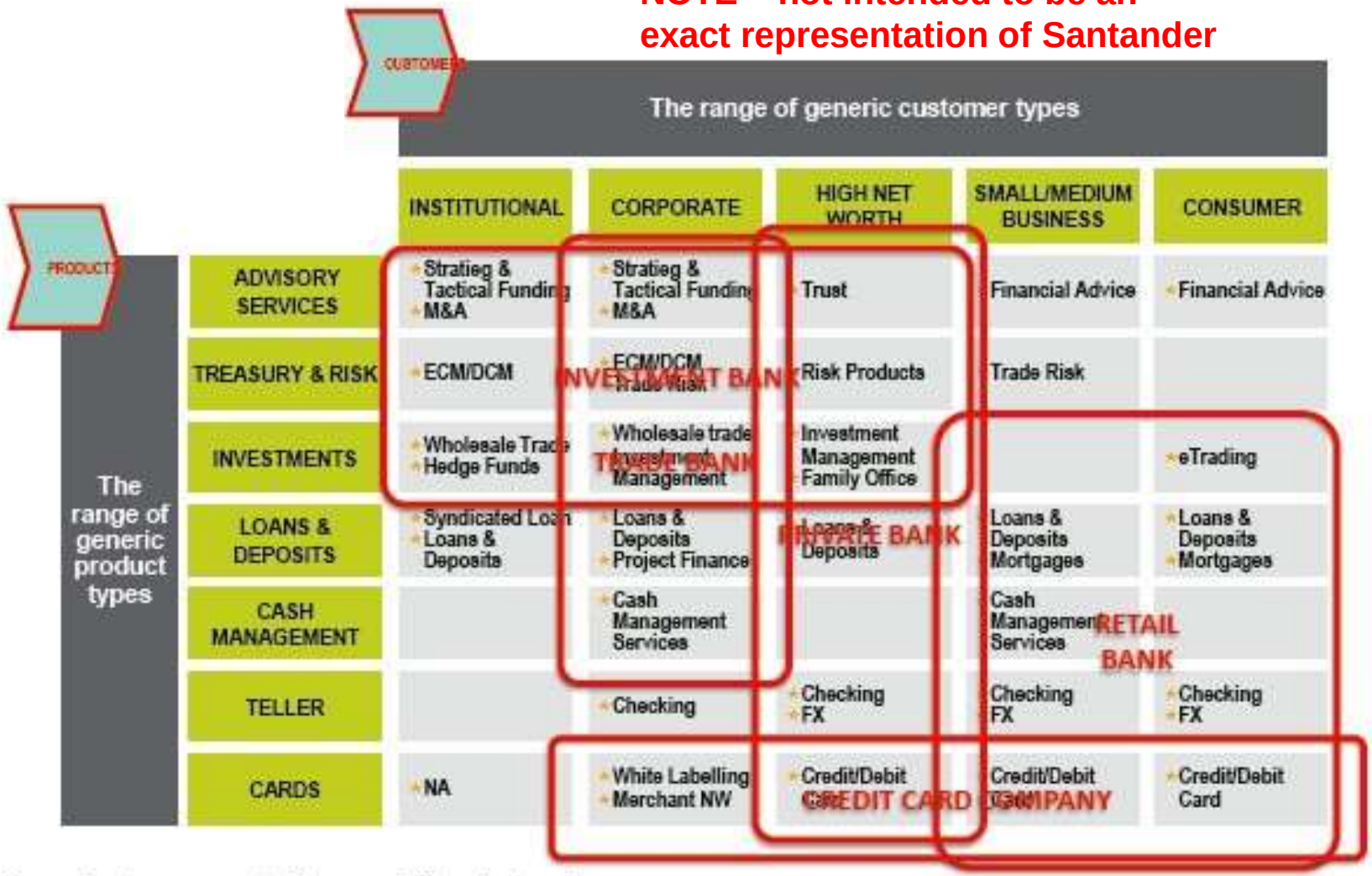
21st October 2020

View of Generic Banking Institution

BIAN Banking Industry Architecture Network

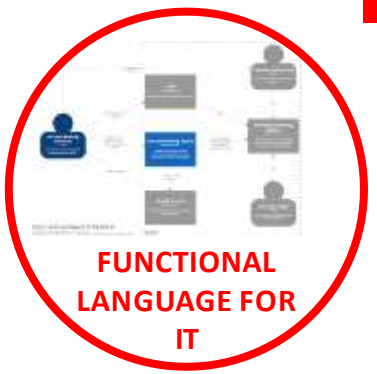
NOTE – not intended to be an exact representation of Santander

Santander is a complex global, multi-entity organization.
For purpose of demonstrating, this is a an example of the generic global bank view across the customer and product dimensions



Examples of product segmentation and "bank type" coverage

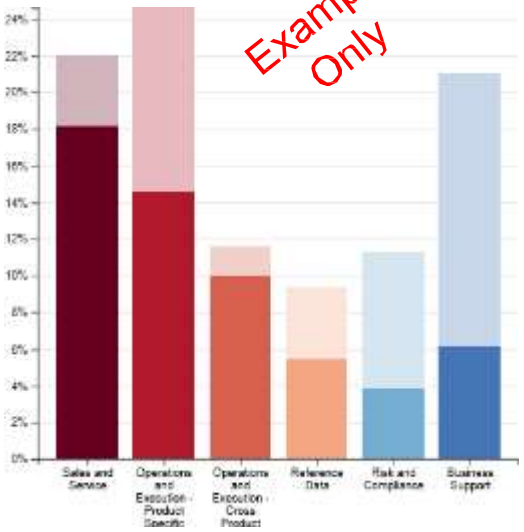
Example BIAN Use Case 1



BIAN AS A FUNCTIONAL LANGUAGE TO ORGANIZE IT LANDSCAPE

- In a complex global, multi-entity organization such as Santander Group, BIAN provides value as a common functional language to organize and manage the Applications Portfolio.
- Key IT portfolio tools are being updated to include BIAN Service Landscape as a new dimension to manage our assets.
- This also provides a view into opportunities such as simplification of the application landscape, reducing costs, decommissioning.

BIAN Coverage



Example BIAN Use Case 2

API DESIGN

Domain Driven Design

BaaS - API FUNCTIONAL REFERENCE FRAMEWORK

- A reference architecture framework has been defined in order to provide common standards across the group.
- API Functions are published in the Intranet API Portal
- Enables more efficient identification of required APIs, also preventing inadvertent duplication.
- Each API is classified under its corresponding BIAN Service Domain.
- API Design: Each API must expose functions from only one BIAN Service Domain.
  - Also recognises, in some cases, a service domain may have more than one API
- An API can delegate responsibility to another API, matching the Service Domain model of delegating responsibilities to other service domains. This is hidden from the consumer.

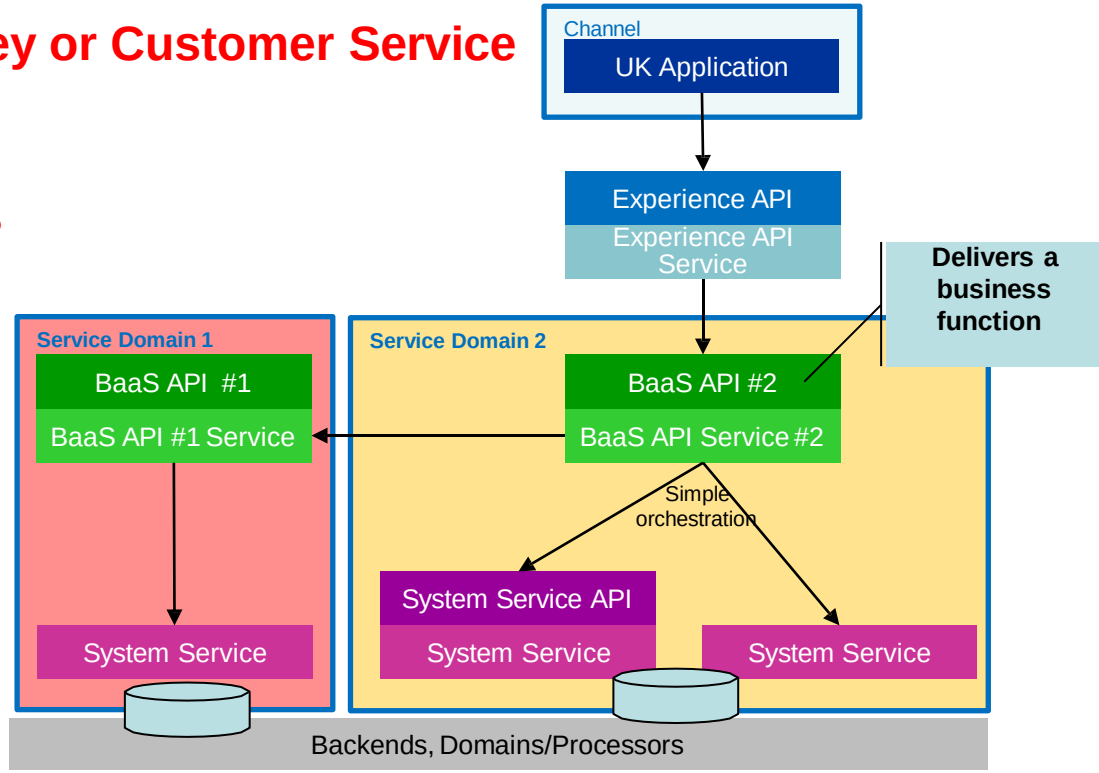
Example BIAN Use Case 2

API DESIGN

Domain Driven Design

BaaS - API FUNCTIONAL REFERENCE FRAMEWORK

- Banking as a Service (BaaS) APIs can be reused and have business logic that is common for many applications according to its BIAN Service Domain (Capability) which sets the logical boundaries and scope of functionality and data (I.e. Domain Driven Design)
- BaaS APIs are Touchpoint, Customer Journey or Customer Service capabilities
- Each BaaS API must exclusively manage it's own data, for the whole lifecycle of that data.



Direction of travel towards service orientated fully compliant landscape

	Type 1. Direct to Core	Type 2. Wrapped Host	Type 3. Component Architecture
Definition	The API routes direct to the core system providing the service. Intermediate channel based access control and 'buffering' is required	Integrating service middleware – a service bus – 'wraps' the host systems. The service bus can offer various host access mitigation capabilities/enhancements	The host services are implemented as loose coupled microservices with complex interactions supported by sophisticated connective middleware
API Service Description	Read only or simple 'atomic' update transactions supported by a single host system. The solution is likely to be host application specific	Enhanced 'simple access' services aligned to established standards. Wrapping may enhance service capabilities and some hosts may support more complex exchanges	Support for flexible and complex interactions involving multiple business activities and processing/decision chains
Examples	◆ Retrieve a balance/account statement ◆ Reference a product/service directory ◆ Initiate a payment	Message conforms to industry standards (e.g. ISO20022) ◆ Retrieve a balance/account statement ◆ Reference a product/service directory ◆ initiate a payment ◆ Customer on-boarding/offers	◆ Prospect on-boarding and origination ◆ Customer dispute/case resolution ◆ Customer relationship development/up-sell/cross-sell campaigns ◆ Third party service integration
Business Drivers	Provide application based access to an established/existing type of customer exchange	Provide application based access with a high degree of standards alignment. Mask/augment host/legacy system limitations.	◆ Support sophisticated interactions ◆ Support new business models ◆ Support for 3rd party integration ◆ Leverage advanced technolgies/architectures

Direct to Core

Wrapped Host

Microservice Architecture

BIAN Aligned

Model Management Practice

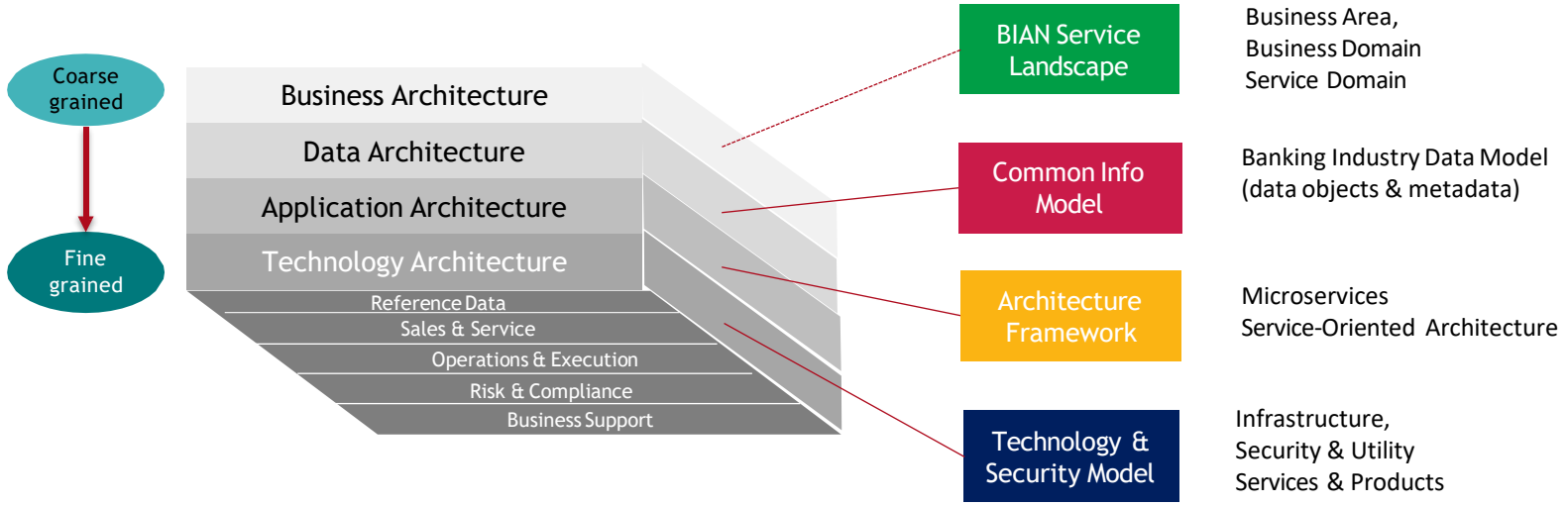
BIAN @ CIBC Overview

Enterprise Architecture

October 22, 2020

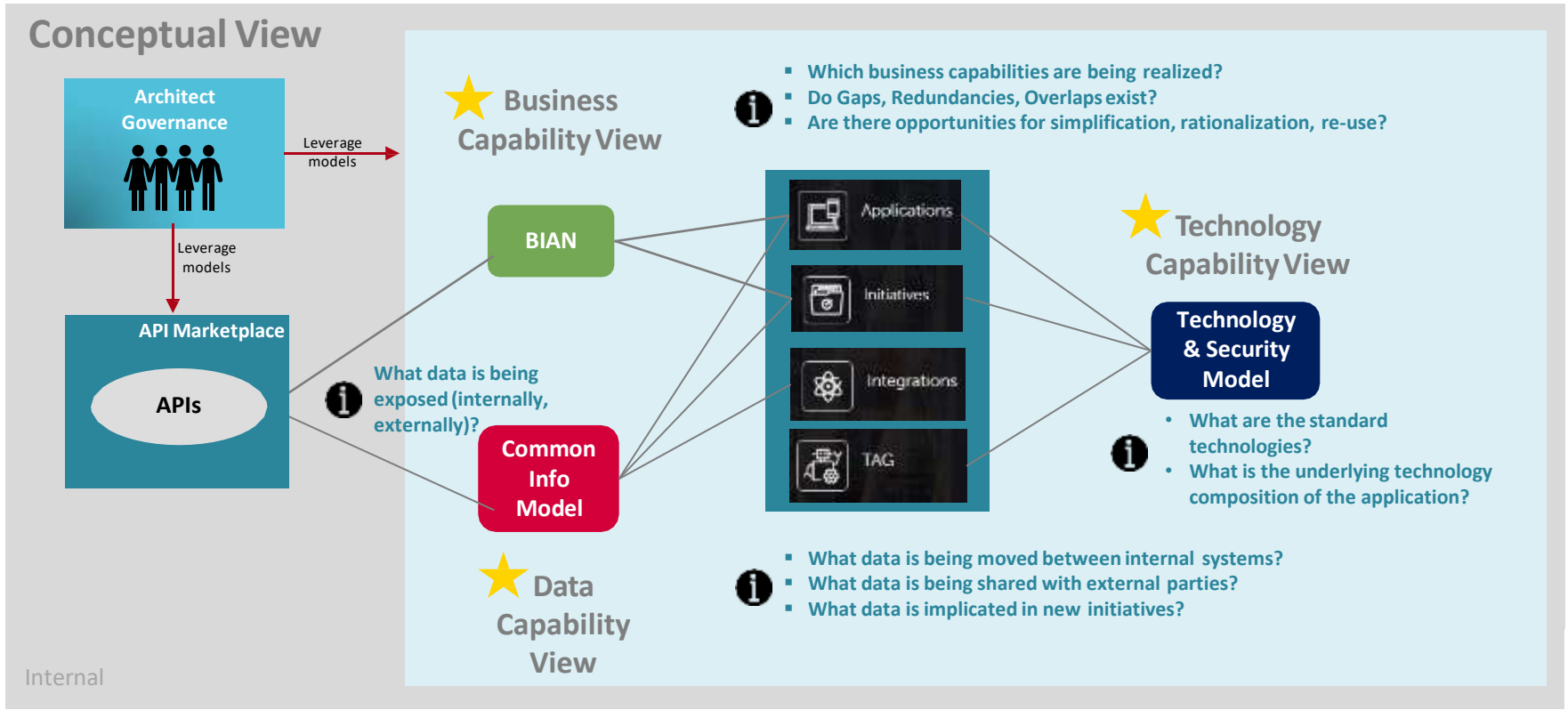
Banking that fits your life.

Our models are framed around the Architecture stack and represents a view to each layer



BIAN is technology and implementation agnostic and focuses solely on the business capability level; capabilities that support the realization of our banking products and services and that help us run as a bank.

Our vision is to leverage models that are fit for purpose and inform our strategic decisions.



BIAN Solution Certification

Certification Working Group, September 2020

BIAN
Banking Industry
Architecture Network

Solution Certification – Overview

Certification Tests

- Definition of Test requirements for each BIAN Artefact Type
  - *Service Domains*
  - *Business Capabilities*
  - *Business Information (BOM, CR, BQ)*
  - *Service Operations / APIs*
  - *Business Scenarios / Wireframes*
- Definition of *Mandatory* v *Optional* requirements.

Application Clusters

For an Application to be certified as BIAN compliant

- Identify relevant BIAN functional building blocks (Service Domains)
- Identify the role of each Service Domain: *Key, Proxy, Utility, External, Peripheral*
- *Identify all BIAN artefacts in scope of the Cluster and apply the relevant Certification Tests*

Status

- ***DRAFTs being reviewed***
  - Product Certification questionnaire (spreadsheet)
  - Product Certification working paper (document).

NEXT

- Run some tests based on Pilot, distinction between self certification & confirmation of the self certification – ***Q1 2021***
- *Produce recommendations for Production testing* ***– Q2 2021***

Pilot agreed

TEMENOS

- ***Temenos Infinity***
  - *Digital Front Office Solution*
  - Greenfield Banking system

DXC.technology

- ***DXC Celeriti***
  - *Enterprise Banking & Lending Solution*
  - Incumbent Core Banking system

Interested Members

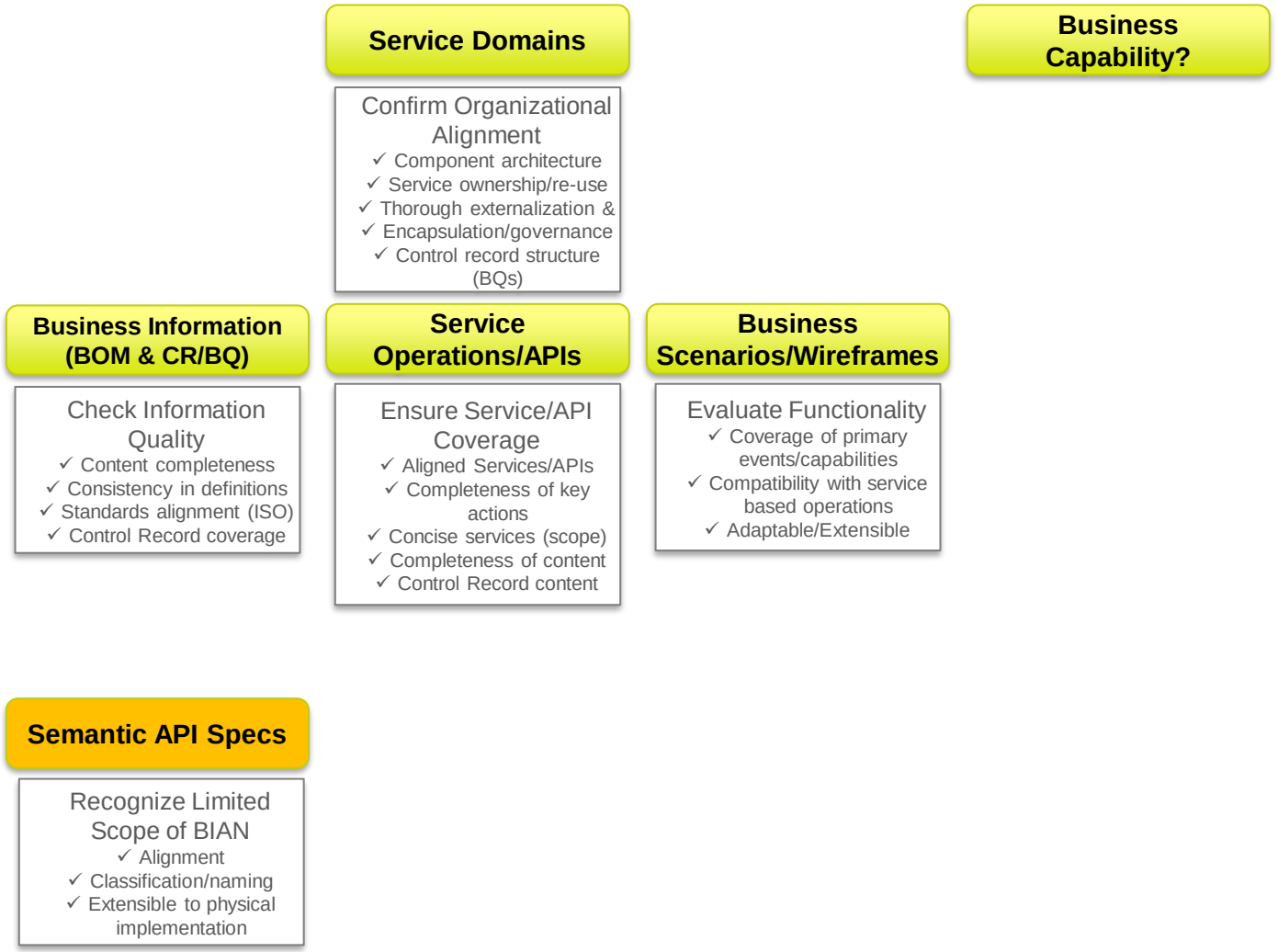
intellect® Design for Digital

UnionBank

Wiki: https://biancoreteam.atlassian.net/wiki/spaces/CER/overview

BIAN

Certification Tests – *Mapped to BIAN Artefacts*



In Progress

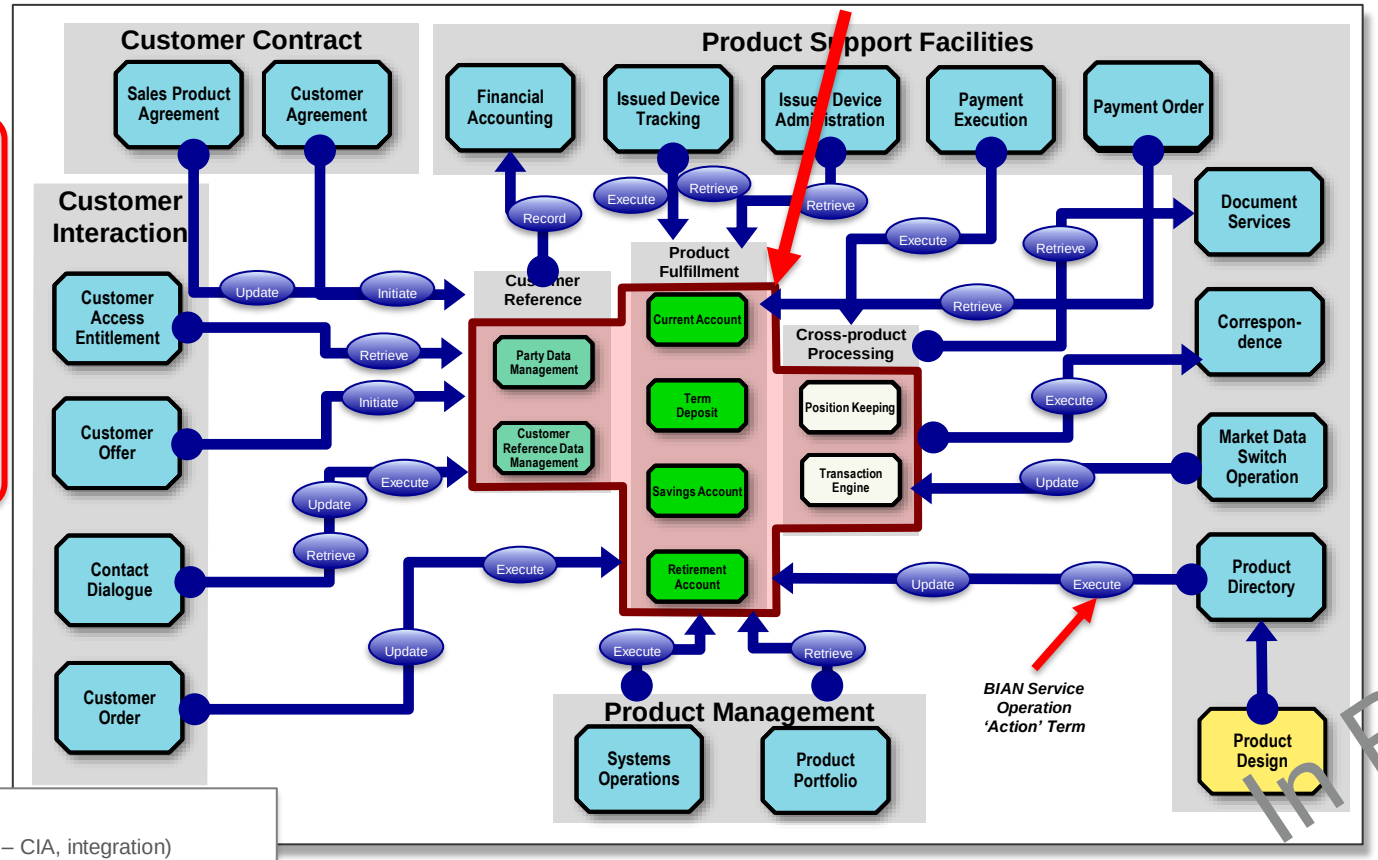
Application Clusters

Example Application Cluster for *Current*, *Savings*, *Retirement & Term Deposit Accounts Offering*
- Including Service Domain Roles

Application Cluster showing Core/Utility/Proxy Roles

Key: Service Domain Roles

- **Service Domain** — **Core = Service Domain wholly contained within cluster**
- **Service Domain** — **Proxy = Local instance synchronised with master**
- **Service Domain** — **Utility = Local instance, no need to synchronise**
- **Service Domain** — **External = First order service operation connections**
- **Service Domain** — **Peripheral = Second order dependencies – for reference**



In Progress

Key Questions

- ✓ Correct/meaningful content (rules – e.g. performance/security XACML – CIA, integration)
- ✓ Clearly defined roles within the system being certified (include Externalization is followed) see guide
- ✓ Map governed information to the scope of the associated control record – check ability to decouple
- ✓ External coordination/synchronization interfaces correctly implemented (including master/proxy coordination)
- ✓ Complete scope of external connections (requires first order connections for transactional & referential dependencies) in order to identify all required Service Operations (checking extracted information content)

BIAN

API Governance Compliance & Feedback

DRAFT – Being looked into

API Governance, Compliance & Feedback

Functionality

- **OAS domains** – Data type compliance eg. currency, DateTime etc.
- **API Standardization** – Out-of-the-box rules
- **Custom rules –** using Regex
- **Webhooks & registry API scripting**

Possible use cases

- Compare implemented APIs against the BIAN standard APIs
  - When members / non-members give us API feedback from real implementations
  - Compliance / Certification
- Provide a starter pack for users, to start a BIAN compliant implementation*

Industry Adoption

- ***Widely used for Swagger JSON / YAML***
- ***Specific Industries***
  - ***SWIFT / ISO20022*** – Payments
  - **HL7 / FIHR** – Healthcare
  - **IATA** – Airline
  - **TMF Forums** – Telecoms
  - **UN Standards**

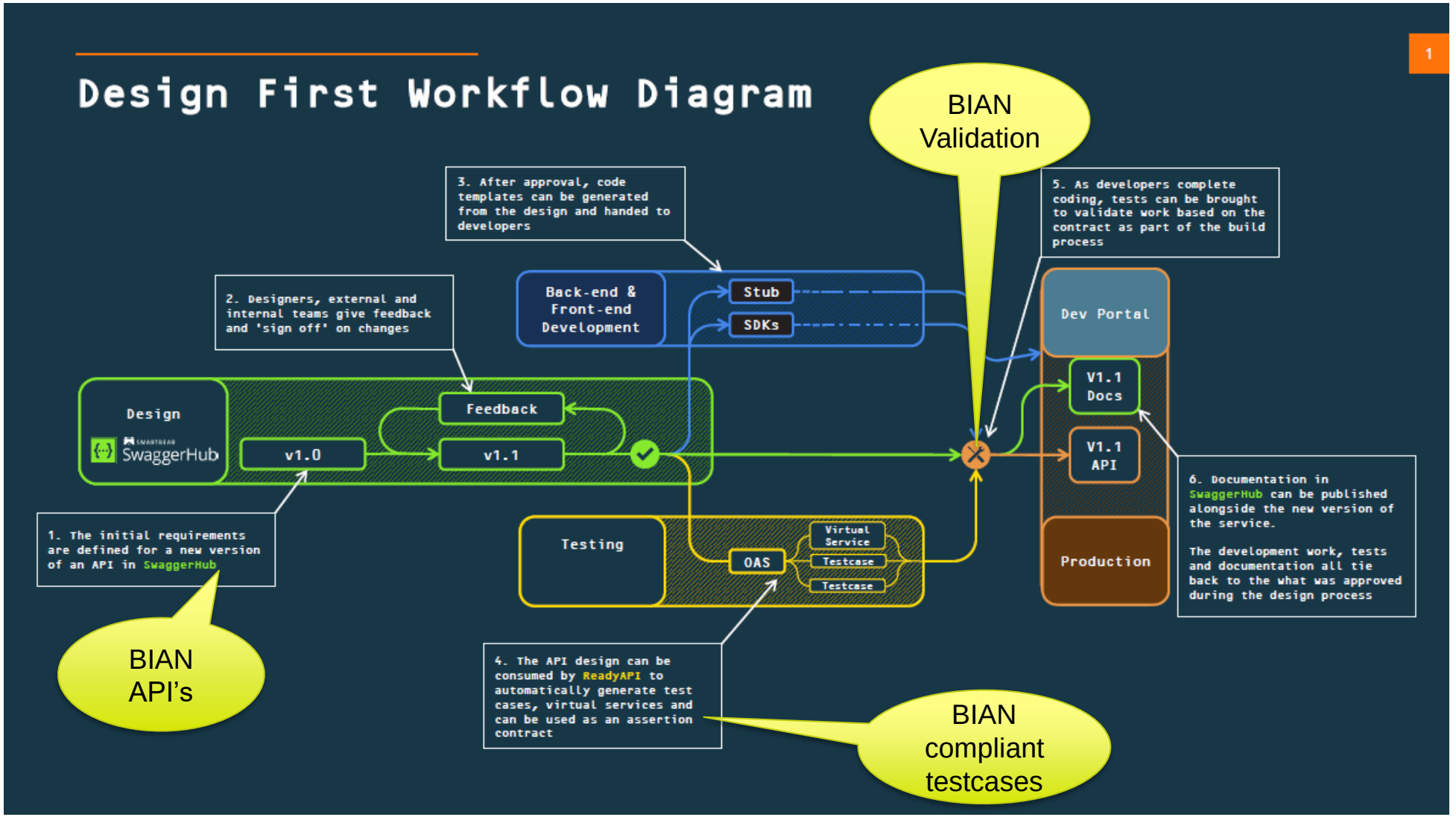
In Progress

DevOps CI/CD SmartBear / Swaggerhub tool

This type of Tooling is already used by:

SWIFT
HR7 (Healthcare)
IATA (Airlines)
UN

In Progress



Thank you

BIAN
Banking Industry
Architecture Network