**Test Plan Approach and**

**LLD – KVP**

IBM Corporation

**Authors**

|  |  |
| --- | --- |
| **Name** | **Role in IBM Project** |
| Sachin | Application Developer |
| Jaya Krishna | Application Developer |
| Siva | Application Developer |

**Document History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Document Revision Description** | **Document Author** |
| 10 July 2024 | 1.0 |  | Sachin, Jaya Krishna, Siva |

# **KVP Context Diagram**

A context diagram provides a high-level view of the system and its interactions with external entities. In this scenario, the system interacts with various external entities to perform the described tasks. Here's a context diagram for the given scenario:

A diagram of a software company

Description automatically generated

# **Process Flow**

**SCENARIO A: User applies for Kisan Vikas Patra (Single holder type) on MB or IB**

1. User will login into the DBP
2. Entry Points:
   * For KVP request, user will have 2 entry points to the KVP Info Page   
     • Entry Point 1: Investment-Account- Add Account-Kisan Vikas Patra   
     • Entry Point 2: Home Page- Invest & Secure-Kisan Vikas Patra
3. Product info page will be displayed, and user can click on Apply Now to proceed.
4. The Following Checks will take place post clicking on Apply Now:   
   • KYC Compliant Check   
   • PAN Availability check   
   • Aadhar Availability check.   
   Note: In case any of the checks fails, journey drops off.
5. Post all the checks, customer lands on KVP landing page.
6. All accounts linked to the CIF will be displayed in the dropdown. User can fund the KVP account from an Individual account, Minor operated by a guardian, Joint account (either or survivor, anyone survivor)
7. User selects the type of KVP account from the drop down:   
   • Single holder type (This will be selected by default)   
   • Joint Type A   
   • Joint Type B   
   Note: In case of Joint Account (Either or survivor), KVP account will be opened in the name of the user who has logged in.
8. If user selects single holder type, then journey follows.
9. User enters the amount to be deposited. Min amount is 1000 and can be in multiples of 100. This amount is configurable and maintained at Admin.
10. Post this maturity amount and tenure is fetched from GBM and displayed.
11. User can enter the nominees, nomination is not mandatory, but the user can add up to 4 nominees. In case the added nominee is a minor, guardian details are also captured.
12. The contribution percentage of nominees is also captured. The total percentage should be 100.
13. Post clicking on Review, review page is displayed.
14. User enters TPIN and authenticates the fund transfer. (Note: Customer account debited- Pool account credited)
15. Post successful TPIN Authentication:   
    • GBM API is called to create an account in GBM.   
    • KVP Account details are pushed against the account created in GBM.   
    • Fund is transferred to KVP Pool account in CBS.
16. Post this, the success screen is displayed to the user.   
    Note: For success screen, DBP flagged-KVP Account number is fetched from GBM and displayed. Deposit ID’s are also DBP flagged in GBM.

**SCENARIO B: User applies for Kisan Vikas Patra (Joint type A or Joint type B) on MB or** **IB**

1. User will login into the DBP
2. For KVP request, user will have 2 entry points to the KVP Info Page • Entry Point 1: Investment-Account- Add Account-Kisan Vikas Patra • Entry Point 2: Home Page- Invest & Secure-Kisan Vikas Patra
3. Product info page will be displayed, and user can click on Apply Now to proceed.
4. The Following Checks will take place post clicking on Apply Now: • KYC Compliant Check • PAN Availability check • Aadhar Availability check. Note: In case any of the checks fails, journey drops off.
5. Post all the checks, customer lands on KVP landing page.
6. All accounts linked to the CIF will be displayed in the dropdown. User can fund the KVP account from an Individual account, Minor operated by a guardian, Joint account (either or survivor, anyone survivor) 7. User selects the type of KVP account from the drop down: • Single holder type • Joint Type A • Joint Type B Page 11 of 27 Classification: Confidential Classification: Confidential
7. If the user selects the Joint type A or Joint type B, then the relevant message to visit branch to continue the journey is displayed.

**KVP - Component Diagram**

A diagram of a service

Description automatically generated

**Use Case Realization - Sequence Diagram**

A screenshot of a computer

Description automatically generated

**For reference, attached is the excel sheet of APIs:**

|  |  |
| --- | --- |
| List of APIs |  |

|  |  |
| --- | --- |
| KVP Domain |  |
| Nominee Domain |  |
| Customer Domain | Customer should have valid Aadhaar and should not be taxpayer. |
| Account Domain | CASA should have Active CASA account. |

**KVP Domain Entity and Domain Validations (Business Validations)**

**KVP Domain Events**

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

**Error Message:**

|  |  |
| --- | --- |
| Error Code | Error Message |
| CUST\_NOT\_FOUND | Scheme is not initialized or empty |
| CUST\_NOT\_INIT | Customer is not available or invalid |
|  |  |

**Data Base Tables**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| TableName | Column Name | Column Comment | Key | Data Type | DATA\_LENGTH |
| investment\_account\_kvp | cif | Customer CIF | K | text | 50 |
| investment\_account\_kvp | application\_form\_number | Application Form Number |  | text | 20 |
| investment\_account\_kvp | deposit\_id | Deposit Id (Transaction Reference Number) |  | text | 20 |
| investment\_account\_kvp | source\_account\_number | CA/SA/OD Account number Select to create Account |  | text | 16 |
| investment\_account\_kvp | source\_account\_name | CA/SA/OD Account Name Select to create Account |  | text | 80 |
| investment\_account\_kvp | date\_of\_opening | Date Of Opening |  | timestamp |  |
| investment\_account\_kvp | account\_name | Name of KVP Account |  | text | 30 |
| investment\_account\_kvp | account\_type | Account Type Single Holder/Joint A / Joint B |  | text | 30 |
| investment\_account\_kvp | interest\_rate | Interest Rate of |  | decimal | 9 |
| investment\_account\_kvp | tenure\_days | Tenure Days |  | int |  |
| investment\_account\_kvp | tenure\_months | Tenure Months |  | int |  |
| investment\_account\_kvp | tenure\_years | Tenure Years |  | int |  |
| investment\_account\_kvp | maturity\_date | Maturity Date Of Account |  | timestamp |  |
| investment\_account\_kvp | maturity\_amount | Maturity Amount Of Account |  | decimal | 20 |
| investment\_account\_kvp | deposit\_amount | Deposit amount |  | decimal | 20 |
| investment\_account\_kvp | nominee\_indicator | Nominee Indicator |  | boolean | 1 |
| investment\_account\_kvp | nominee\_detail | Nominee Details | list <FROZEN<nominee\_info>> | nominee\_info | UDT |
| investment\_account\_kvp | declaration\_indicator | Self-declaration for the mandatory government guidelines - Account Declaration (true/false) |  | boolean | 1 |
| investment\_account\_kvp | declaration\_date | Account Declaration Date |  | timestamp |  |
| investment\_account\_kvp | notification\_indicator | Account Open/Closing Notification Indicator true/false |  | boolean | 1 |
| investment\_account\_kvp | principal\_amount | Principal Amount Of Account |  | decimal | 20 |
| investment\_account\_kvp | interest\_amount | Interest Amount Of Account |  | decimal | 20 |
| investment\_account\_kvp | closure\_credit\_account\_number | CA/SA/OD Account number Select to Close Account |  | text | 16 |
| investment\_account\_kvp | closure\_credit\_account\_name | CA/SA/OD Account Name Select to Close Account |  | text | 80 |
| investment\_account\_kvp | email\_id | Email Id |  | text | 80 |
| investment\_account\_kvp | mobile\_number | Mobile Number |  | text | 20 |
| investment\_account\_kvp | closing\_declaration\_indicator | Self-declaration for the mandatory government guidelines - Clsoing Account Declaration (true/false) |  | boolean | 1 |
| investment\_account\_kvp | closing\_declaration\_date | Account Closing Declaration Date |  | timestamp |  |
| investment\_account\_kvp | date\_of\_closing | Date Of Opening |  | timestamp |  |
| investment\_account\_kvp | closing\_mode\_of\_transfer | Closing Payment Mode of Transfer ( Account Trasnfer or Demand Draft ) |  | text | 30 |
| investment\_account\_kvp | total\_amount\_payable | Closing Payment Payable |  | decimal | 20 |
| investment\_account\_kvp | status | Status Of an Account Creation ( Complete/Incomplete/Closed/Withdrawal) |  | text | 20 |
| investment\_account\_kvp | branch\_code | Branch Code |  | text | 6 |
| investment\_account\_kvp | branch\_sol\_id | Sol Id of the Branch |  | text | 8 |
| investment\_account\_kvp | branch\_name | Name of the Branch |  | text | 30 |
| investment\_account\_kvp | branch\_addressline1 | Address Line1 of Branch |  | text | 45 |
| investment\_account\_kvp | branch\_addressline2 | Address Line2 of Branch |  | text | 45 |
| investment\_account\_kvp | branch\_addressline3 | Address Line3 of Branch |  | text | 45 |
| investment\_account\_kvp | branch\_addressline4 | Address Line4 of Branch |  | text | 45 |
| investment\_account\_kvp | branch\_address\_pincode | PinCode |  | text | 10 |
| investment\_account\_kvp | branch\_postoffice | PostOffice Name of the Branch |  | text | 45 |
| investment\_account\_kvp | branch\_city\_code | City Code |  | text | 20 |
| investment\_account\_kvp | branch\_city | City |  | text | 45 |
| investment\_account\_kvp | branch\_district\_code | District Code |  | text | 20 |
| investment\_account\_kvp | branch\_district | District of Branch |  | text | 45 |
| investment\_account\_kvp | branch\_sub\_district | Sub District of Branch |  | text | 45 |
| investment\_account\_kvp | branch\_state\_code | State Code |  | text | 20 |
| investment\_account\_kvp | branch\_state | State of Branch |  | text | 30 |
| investment\_account\_kvp | branch\_working\_hrs | Working Hrs of branch (10:00am to 4:00pm) |  | text | 50 |
| investment\_account\_kvp | branch\_working\_days | Working Days of branch (Mon to Sat,1st,3rd and 5th Sat) |  | text | 50 |
| investment\_account\_kvp | active\_flag | Active Flag, Record is active or logically deactivated true/false |  | boolean | 1 |
| investment\_account\_kvp | created\_date | Date on which record is Inserted |  | timestamp |  |
| investment\_account\_kvp | created\_by | Job/ETL/Service name through which record is Inserted |  | text | 50 |
| investment\_account\_kvp | modified\_date | Date on which record is Modified |  | timestamp |  |
| investment\_account\_kvp | modified\_by | Job/ETL/Service name through which record is Modified |  | text | 50 |
| investment\_account\_kvp | channel | Channel Name ( DBP,DBP-MB-MSME(Mobile Banking (MSME App)),DBP-MB-RETAIL(Mobile Banking (Retail App)),DBP-Net Banking,Open Web, DBP-Assisted Portal,Whatsapp,Conv AI,Wearables,Others – Specify) |  | text | 20 |

**Application Properties**

|  |  |  |  |
| --- | --- | --- | --- |
| Property Name | Short Description | Property |  |
| **Apicurio** | It is to be used for validating inputs (Json Schema) received |  |  |
| **RedHat** | These configuration properties are used to specify the connection details for your application to interact with a Red Hat Data Grid instance. This could involve reading from or writing to the data grid, depending on the use case. The properties include the host address, port number, and credentials (username and password) needed for authentication and communication with the data grid |  |  |
| **Swagger** | It is used to customize the path at which the Swagger UI is accessible in a Spring Boot application. This configuration property is related to the integration of Springdoc Open API with Spring Boot. | springdoc.swagger-ui.path=/swagger-ui.html |  |
| **Path** | It is used to provide deployed path to call API's |  |  |
| **API Endpoints** | These endpoints are called to get proxy data |  |  |

|  |  |  |
| --- | --- | --- |
| **KVP Query** |  |  |
| Property Name | Short Description | Property |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Database connection** | These properties are used to connect database i.e cassandra | quarkus.cassandra.contact-points=172.29.24.144:9042 quarkus.cassandra.local-datacenter=bangaloredc1 quarkus.cassandra.auth.username=cassandra quarkus.cassandra.auth.password=cassandra |  |  |
| **Proxy calls** | These are used to call proxy api's and calling varions common Query Services |  |  |  |
| **Email** | These are related to generation of e-mail. | email.subject="Email Statement for your Bank Account" quarkus.tls.trust-all=true quarkus.index-dependency.emailclient.group-id=com.ubi.dbp.email.client quarkus.index-dependency.emailclient.artifact-id=email-service-client |  |  |

**KVP Command Service**

|  |  |  |
| --- | --- | --- |
| Property Name | Short Description | Property |
| **Database connection** | These properties are used to connect database i.e. Cassandra | quarkus.cassandra.contact-points=172.29.24.144:9042 quarkus.cassandra.local-datacenter=bangaloredc1 quarkus.cassandra.auth.username=cassandra quarkus.cassandra.auth.password=cassandra |
| **Proxy calls** | These are used to call proxy API’s and calling various common Query Services |  |
|  |  |  |
| **Kafka Details** | These are used to send and consume message |  |
|  |  |  |
| **Config Map** | Config map is an API object which is mainly used to store non-confidential data |  |

1. KVP:

KVP use case will perform the below steps:

The pattern we are using for KVP development is as follows:

**I. KVP BFF Microservices**

* Spring Boot API is utilized for BFF calls and provided to the Frontend team for UI migration.
* The API will consume and fetch from the query microservice developed for read calls.
* Caching is implemented at the BFF level for the required APIs.
* A REST client is used to consume the query service.
* GSON will be the media type for response handling.

**ii. KVP Query Microservices**

* Query microservices are developed using Spring Boot.
* The API will fetch data from CBS based on the required endpoint.
* A proxy is designed for GBM migration to be called from query microservices.
* GET/READ API microservices are designed in Spring Boot.
* GSON will be the media type for response handling.

**iii. KVP Command Microservices**

* Commands are developed for save/open/close APIs for KVP.
* Spring Boot is used for developing the command service.
* Journey orchestrator is utilized for managing the journey progress.
* Kafka is used for publishing and subscribing to the journey.

**iv. CBS Proxy**

* A proxy is developed for communication with the CBS.
* Spring Boot is used for the development of the proxy.
* Read/Write APIs are provided to fetch and store data from CBS.

**v**. **GBM Proxy**

* A proxy is developed for communication with the GBM.
* Spring Boot is used for the development of the proxy.
* Read/Write APIs are provided to fetch and store data from GBM.

1. **Domain and Event Models**

**Entities:**

An entity is an object with a unique identity that persists over time. For instance, in a banking application, customers and accounts would qualify as entities. Each entity possesses a distinct identifier within the system, facilitating retrieval or lookup. However, this identifier may not necessarily be directly exposed to users and can take the form of a GUID or a primary key in a database. Additionally, an entity's identity may extend across multiple bounded contexts and endure beyond the application's lifecycle. For example, bank account numbers or government-issued IDs remain independent of any specific application's duration. Furthermore, the attributes associated with an entity may undergo changes over time, such as a person's name or address, while still representing the same individual. Entities are capable of holding references to other entities.

**Value objects:**

Unlike entities, a value object lacks identity and is solely defined by the values of its attributes. These objects are immutable, meaning any updates necessitate the creation of a new instance to replace the previous one. While value objects can incorporate methods encapsulating domain logic, these methods should not produce side effects on the object's state. Common examples of value objects include colors, dates and times, and currency values.

**Aggregates:**

An aggregate delineates a consistency boundary around one or more entities. Within an aggregate, precisely one entity serves as the root, facilitating lookup through its identifier. Other entities within the aggregate are regarded as children of the root and are referenced through pointers from the root. Aggregates serve the purpose of modeling transactional invariants. Real-world scenarios often entail intricate webs of relationships—customers create orders, orders contain products, and products have suppliers. In such cases, ensuring consistency across multiple related objects poses a challenge. Aggregates address this challenge by providing a means to manage and enforce transactional invariants, thus guaranteeing consistency within the domain.

1. **Deployment Model**

**Application Container (Module-based Application):**

Embracing an application container model facilitates the deployment of modularized applications. By encapsulating various modules within containers, each component can be managed independently, promoting scalability, flexibility, and ease of maintenance. This approach enables seamless deployment and orchestration of application components, facilitating rapid development and deployment cycles.

**Topics (Kafka Integration):**

Leveraging Kafka for topic management enhances communication and data streaming within the application ecosystem. Topics serve as communication channels that enable producers to publish messages and consumers to subscribe and process them. By utilizing Kafka's robust messaging system, applications can achieve real-time data processing, fault tolerance, and scalability. This integration empowers applications to efficiently handle large volumes of data and enables event-driven architectures, fostering responsiveness and agility in application development and deployment.

1. **DevSecOps Considerations and Configurations**

DevOps is a software development approach that emphasizes collaboration, integration, automation, and continuous delivery. In this context:

**Package (Maven Integration):**

Utilizing Maven for packaging and building allows for efficient management of project dependencies and the creation of deployable artifacts. Maven streamlines the build process by automating tasks such as compilation, testing, and packaging, enhancing the speed and reliability of software delivery.

**Container Image (Building):**

Building container images enables the encapsulation of applications and their dependencies into portable, scalable units. Through this process, developers can create consistent environments across various platforms, making deployment more predictable and manageable.

**Tekton Triggers (Webhook Integration):**

Integrating Tekton triggers via webhooks facilitates automation in the CI/CD pipeline. Webhooks provide a mechanism for triggering Tekton pipelines based on specific events, such as code commits or pull requests. This automation streamlines the development workflow, allowing for faster feedback loops and smoother delivery cycles.

**ArgoCD Configuration (Pull-Based Mechanism):**

Implementing ArgoCD with a pull-based mechanism simplifies and automates the deployment of applications in Kubernetes environments. ArgoCD continuously monitors a Git repository for changes to application configurations and automatically synchronizes the desired state with the actual state of the cluster. This approach ensures consistency and reliability in application deployment while minimizing manual intervention.