Document Purpose

This document is designed to provide a comprehensive understanding of the network architecture integrated with various technical components. It aims to help team members gain a clear insight into the functionalities of each component, ensuring they can effectively contribute to the project. By detailing the design, setup, and interactions of the network, this guide serves as a valuable resource for understanding how the different elements work together to achieve the desired outcomes.

Document Scope

This document offers a comprehensive overview of the API integration and network architecture, detailing the integrity and interactions of all technical components involved in the project.

Audience of the Document

This document is intended for software professionals—including programmers, testers, system architects, and IT support staff—who are involved in the design, development, maintenance, and testing of the SBI ePay 2.0 system. The purpose of this document is to provide a comprehensive overview of the network architecture underlying SBI ePay 2.0, a leading payment gateway solution that facilitates secure and seamless digital transactions.

Application Architecture of SBI ePay 2.0

The document explains the key components of the network architecture, the roles they play in ensuring the system's scalability, security, and availability, and how these components interact to support high-performance payment processing. Whether you are working on the back-end integration, testing payment flows, or analyzing the system's performance, this document will serve as a valuable resource to help you understand the technical intricacies of SBI ePay 2.0's network structure.

By understanding the network architecture of SBI ePay 2.0, software professionals can effectively contribute to the system's development, integration, testing, and maintenance. The modular design, coupled with a focus on security and scalability, ensures that the platform can meet the growing demands of online payment processing in a secure and efficient manner.

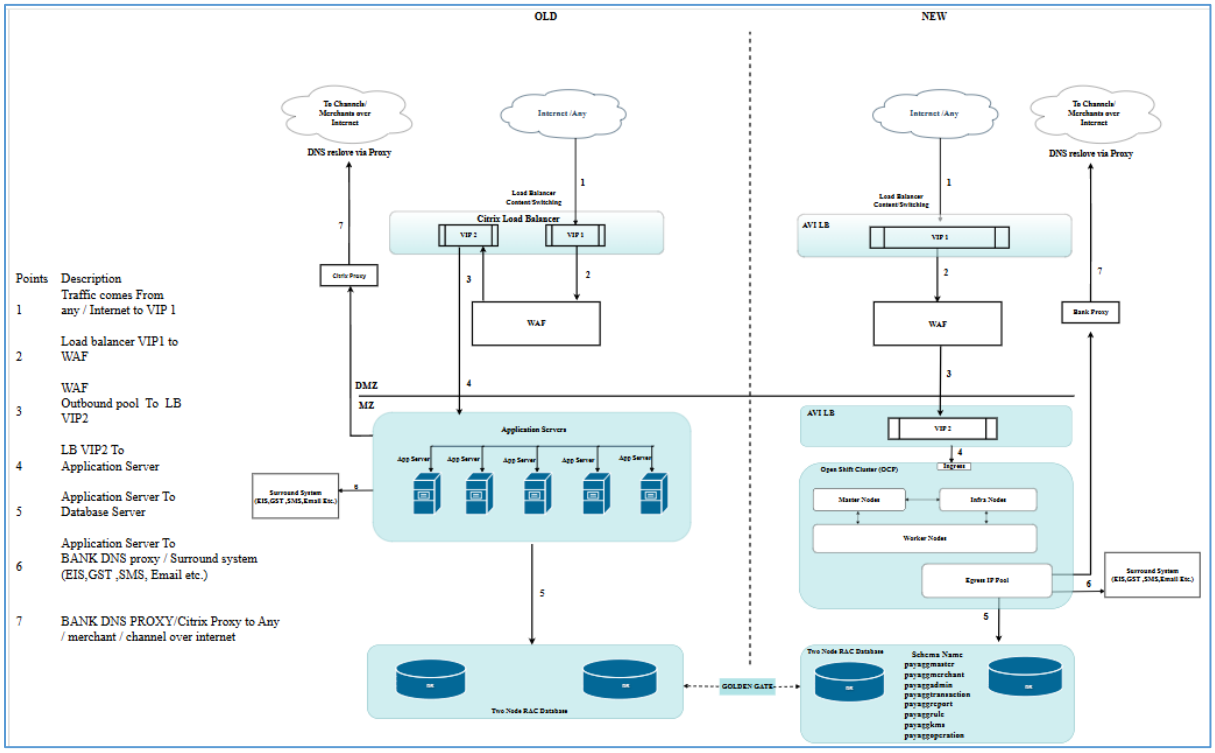
SBI ePay 2.0 Network Architecture

SBI ePay 2.0's network architecture is built to provide a robust, scalable, and secure platform for processing online payments. The system follows a multi-layered approach, utilizing cutting-edge technologies and frameworks to manage the various demands of modern payment processing, including real-time transactions, load balancing, and data security.

Architectural, OCP, and Network diagram Descriptions are as follows:

Kindly relate each diagram with the *High-Level Network Diagram (Old and New OCP Environments)*

High-Level Network Diagram (Old and New OCP Environments)



OLD OCP Environment

Traffic comes from the Internet, and lands at physical devices – Citrix Load Balancer.

VIP1 is used for Incoming Traffic, whereas VIP2 is a Citrix proxy used as a forward proxy.

VIP1 is for managing and filtering internet traffic, whereas VIP2 is for communication.

Bank Proxy will act as a firewall where traffic will be managed.

The Internal Application Server will communicate with Databases and the surround systems which will further distribute the traffic to the various channels.

New OCP Environment

Traffic Flow and Network Architecture in the New OCP Environment:

In the New OCP Environment, traffic from the external internet is directly routed to the WAF (Web Application Firewall), which is the bank's security device. Unlike the previous setup, there are no Citrix devices involved in this flow. The WAF acts as the first line of defence, filtering and securing incoming traffic.

From the WAF, traffic is forwarded to the Software Load Balancer (AVI LB). The AVI Load Balancer operates in a two-layer architecture:

- The upper layer, known as the DMZ (Demilitarized Zone), handles traffic that interacts with external sources.
- The lower layer, called the MZ (Management Zone), is where traffic is directed after passing through the DMZ for further processing.

Once traffic reaches the AVI LB, it is directed to an internal application server. This server processes traffic across multiple layers:

- Master Nodes: Responsible for orchestrating tasks.
- Infra Nodes: Handle infrastructure management.
- Worker Nodes: Perform the actual processing and task execution.

From the internal application server, traffic is sent to the Database and Surround Systems for storage and processing. This communication ensures that internal systems are continuously updated and can perform required operations efficiently.

In contrast to the old environment, the traffic and data from the internal systems are now forwarded by the Bank Proxy Server to various channels for further distribution to their respective destinations. This improves the scalability and management of traffic as it moves through different layers of the architecture.

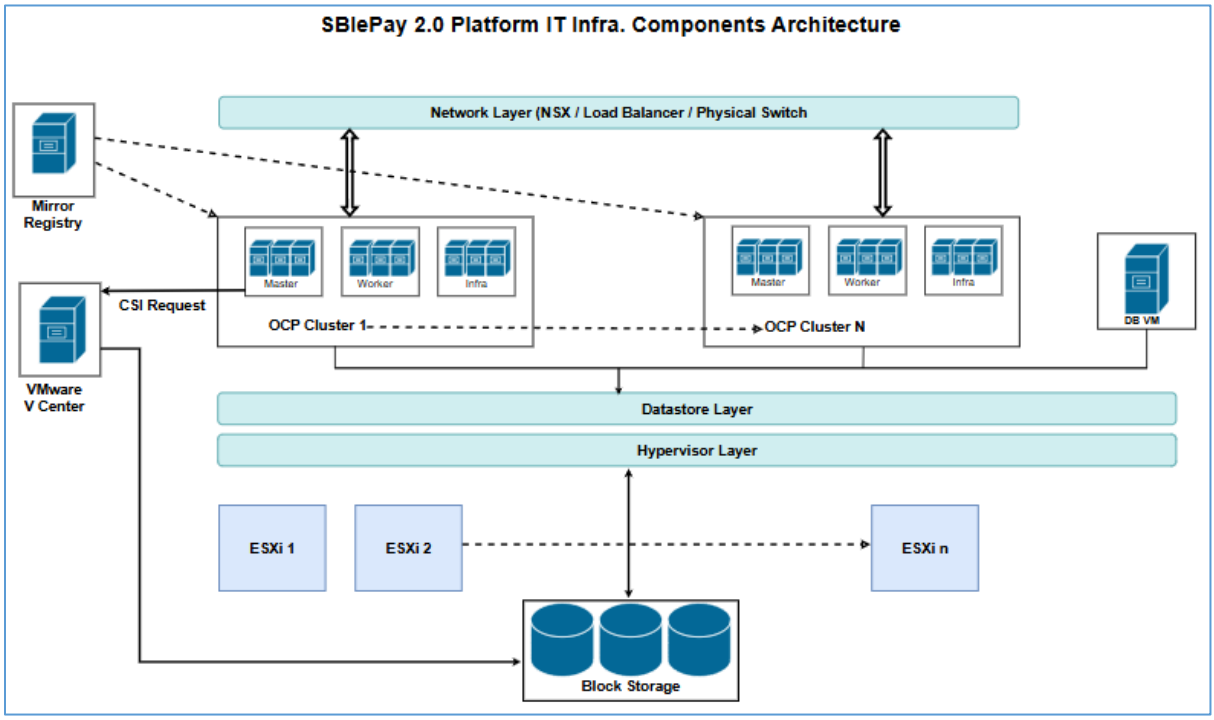
Key Differences Between Old and New Environments:

- In the old environment, the setup included 2-node clusters for handling traffic and processing.
- In the new environment, the architecture is more scalable and segmented, with 3-node clusters and 8 database instances for improved performance, redundancy, and fault tolerance.

The Old and New environments are connected by a bridge called Golden Gate, which enables data migration and synchronization between the two. The Golden Gate ensures that data from the old environment is seamlessly transferred to the new one, ensuring continuity and minimal disruption during the migration process.

This version is more concise, with clear breakdowns of each component's function and differences between the old and new environments. It also improves readability and flow, making it easier to understand the transition and system architecture.

IT Infrastructure Components Architecture



The number of OCP clusters are processing the incoming data traffic through various layers such as the Network layer, Datastore layer, and hypervisor layer.

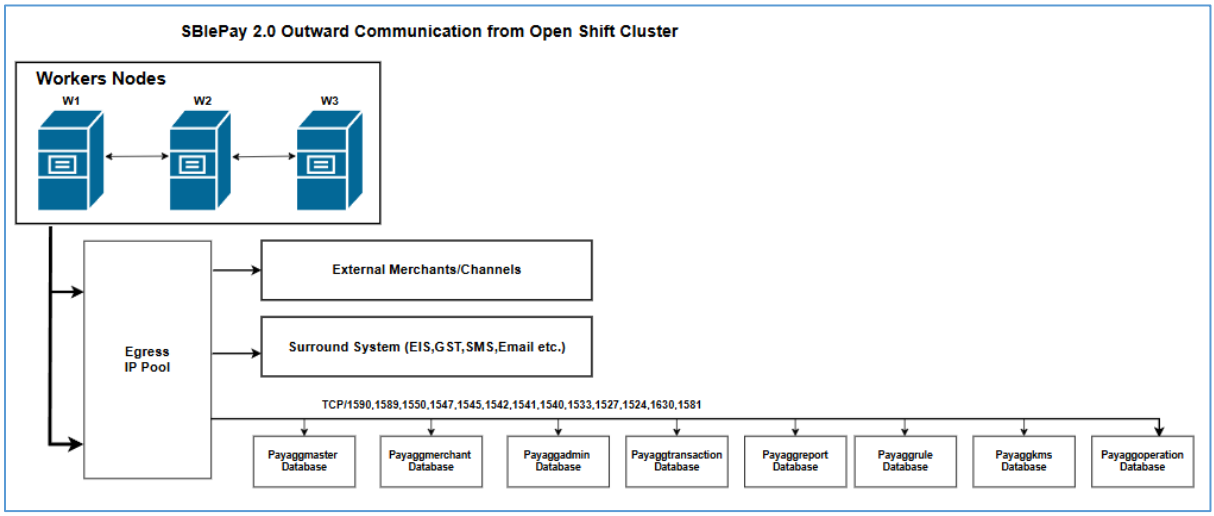
The number of virtual machines is configured using the Datastore layer naf hypervisor layer, a service that is controlled by Meghdoot.

The above-configured machines and the block storages are managed by VMWare V Center.

The n number of configured clusters (c1, c2, c3,….Cn) are managed by Mirror Registry. The entire graphical view and images will be managed by the Mirror Registry.

==========

SBIePay 2.0 Outward Communication from Open Shift Cluster

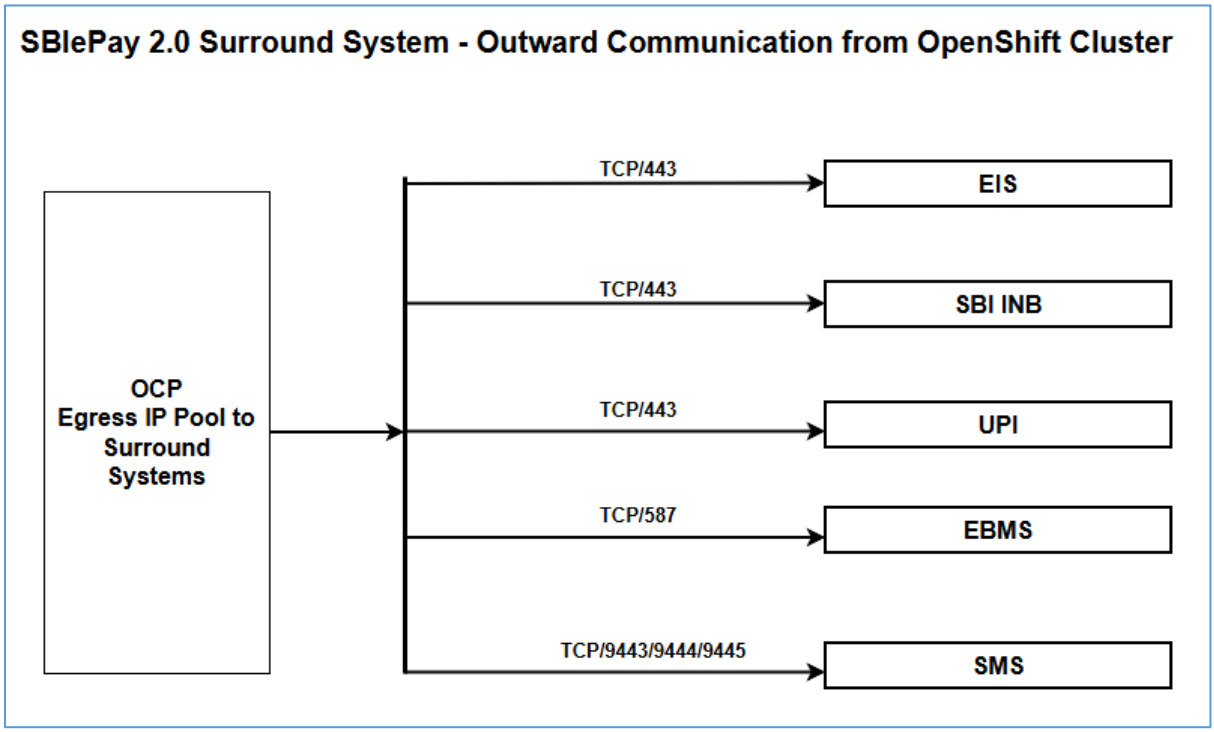


In the segregated cluster, there are three types of nodes, namely, Master Node, Infra Node, and Worker Node, and each is available in quantity 3.

While dealing with internal processing, the Worker Nodes are in communication with the Egress IP Pool.

This Egress IP Pool further communicates with External Merchants/Channels, Surround Systems (EIS, GST, SMS, Email), and databases, 8 in number, each dedicated to its respective storage.

SBIePay 2.0 Surround System to Outward Communication from Open Shift Cluster

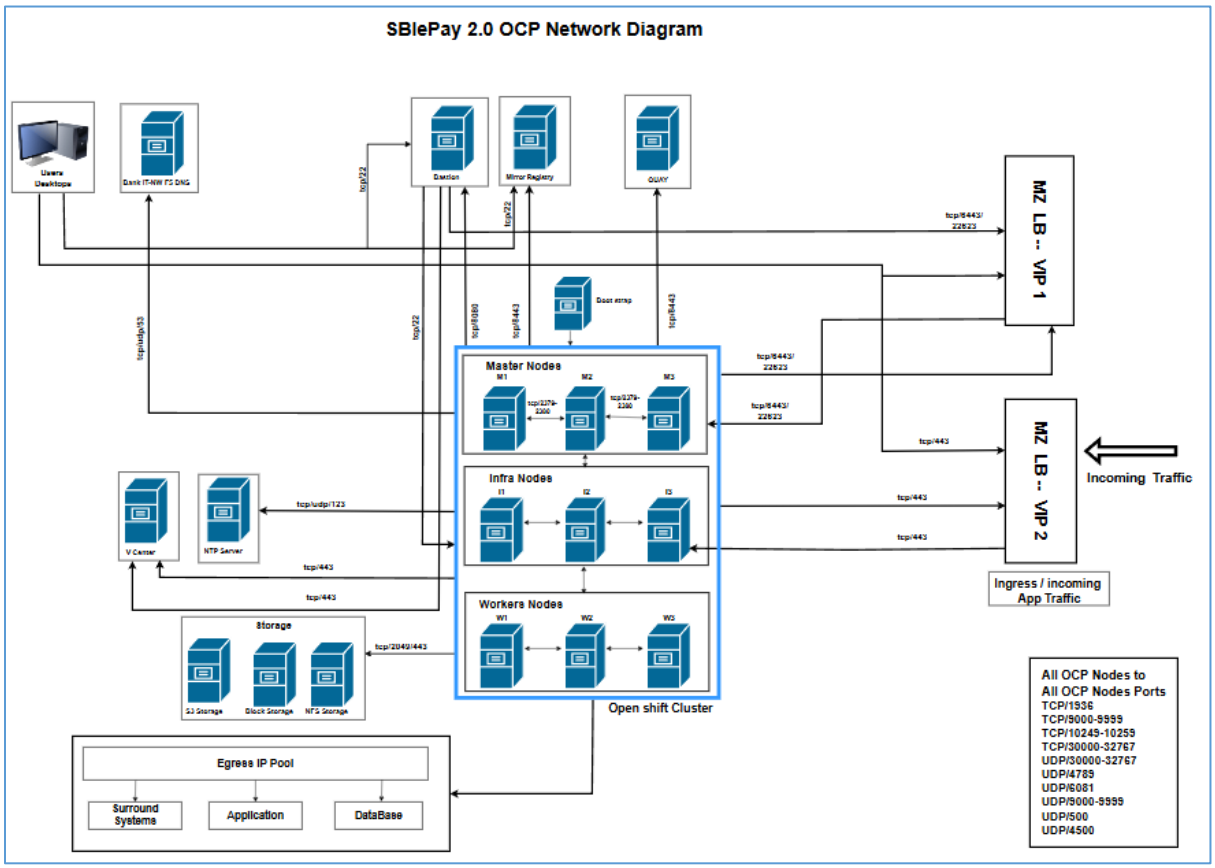


From Egress IP Pool to Surround systems, communication takes place with the following TCP Ports:

TCP Port Number	Channel / Service
TCP/443	Dedicated to EIS for the Loan purpose
TCP/443	Dedicated to SBI INB for SBI Internet Banking
TCP/443	Dedicated UPI for SBI UPI
TCP/587	Dedicated to EBMS for SBI Email services
TCP/9443/9444/9445	Dedicated to SMS for SBI SMS services

SBI ePay 2.0 OCP Network Diagram

Kindly relate each diagram with the *High-Level Network Diagram (Old and New OCP Environments)*



As discussed before, in the segregated cluster, there are three types of nodes, namely, Master Node, Infra Node, and Worker Node, and each is available in quantity 3.

For the configuration of the above cluster, WAF, VIP1, VIP2, Database servers, and Egress IP Pool are used collaboratively, having dedicated responsibility for each component.

WAF – The external internet traffic will directly be landed at WAF through VIP1.

VIP2 is responsible for transactions and incoming traffic.

VIP1 is responsible for managing clusters.

The Egress IP Pool is for internal processing. Infra node virtual machines will be managed by Egress services, at which point internet traffic will be landed.

Internal cluster processing will be processing and communicating with the NTP Server and V Center.

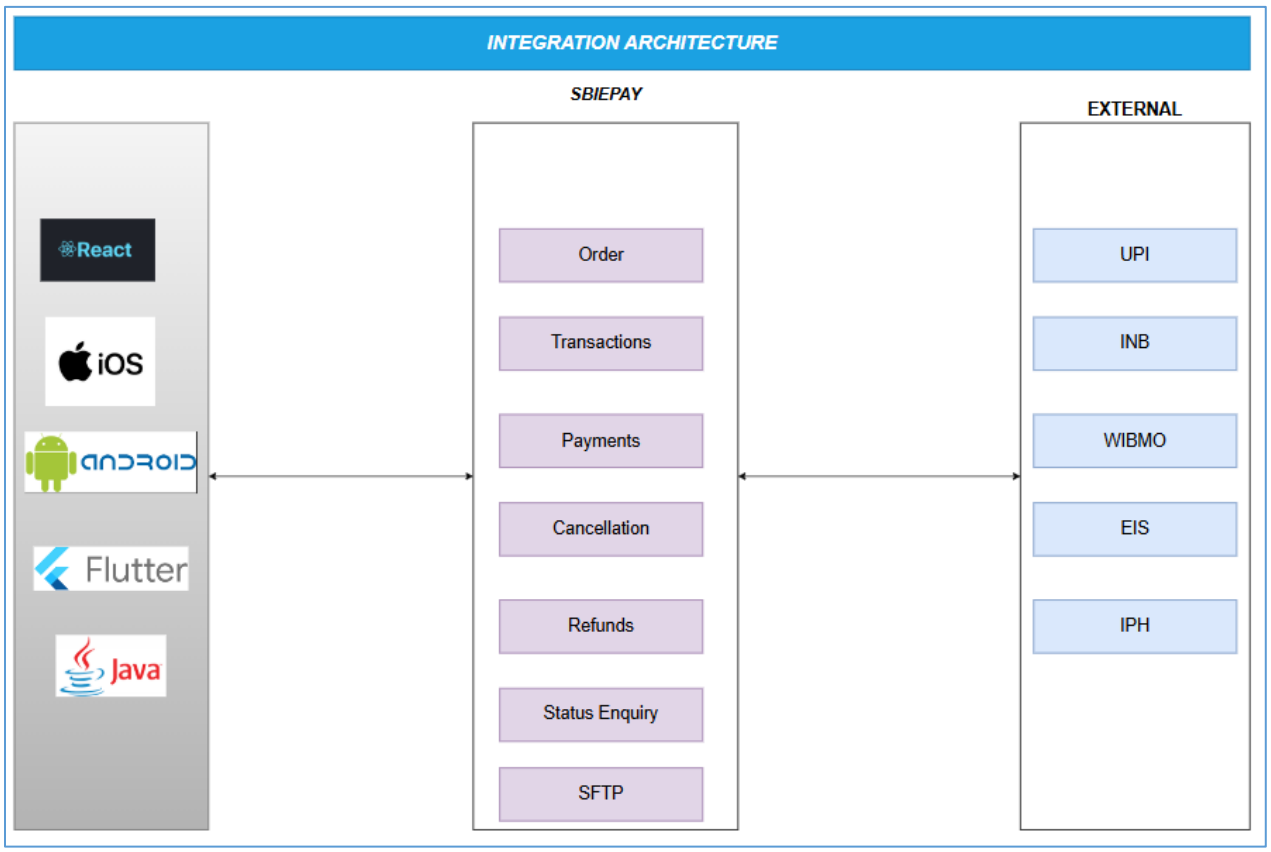
V Center is configured to check the cluster's graphical mode.

SBI ePay 2.0 Application Architecture

Application Architecture diagram Descriptions are as follows:

SBI ePay 2.0 Integration Architecture

Kindly relate each diagram with the architectural diagram *integration_architeture.drawio.html*



During any transaction, the first call will be from UI Frontend, calling the respective service.

Based on the technology used, accordingly, respective service API will be called.

For example,

As we can see in the above diagram, the frontend UI will be using different technologies to access the respective SBIEPAY services:

Java and React by Java technology will be calling all the SBIEPAY services as and when required.

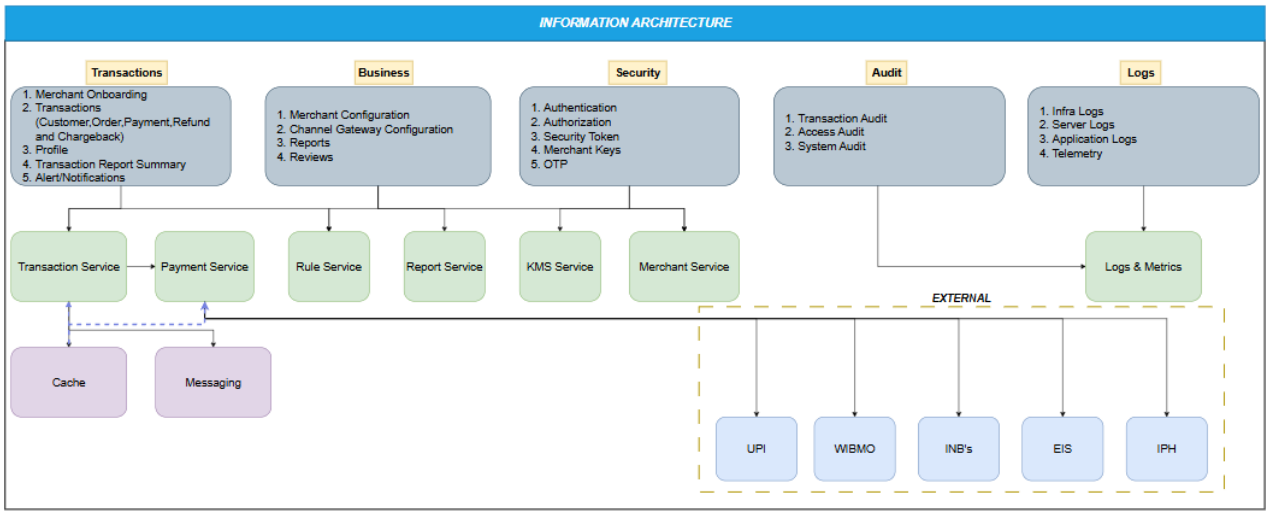
Similarly, iOS will be using the Flutter and iOS technologies to call the SBIEPAY services.

Android will be used by the Android platform users to call SBIEPAY services.

All the above are different platforms using which the SBIEPAY services will be used.

Once the SBIEPAY services started interacting with frontend, external third party services will be used for further part of transaction processing.

SBI ePay 2.0 Information Architecture



The above diagram explains the business functionalities that use its respective services for transaction processing:

The functionalities listed below will call the Transaction Service API and Payment Service API for its processing:

- Merchant Onboarding
- Transactions
- Profile
- Transaction Report Summary
- Alert/Notifications

These Transaction Service and Payment Service APIs will interact with the Cache server and the notification services during the transaction processing.

Also, all the following third-party external services will be called by Payment Service APIs during the payment processing:

- UPI
- WIMBO
- INB’s
- EIS
- IPH

The functionalities listed below will call the Business Service API for its processing:

- Merchant Configuration
- Channel Gateway Configuration
- Reports
- Reviews

This Business Service API will interact with the Rule service and the Report Service during transaction processing.

The functionalities listed below will call the Security Service API for its processing:

- Authentication
- Authorization
- Security Token
- Merchant Keys
- OTP

This Security API will interact with the KMS service and the Merchant Report Service during transaction processing.

The functionalities listed below will call the Audit Service API for its processing:

- Transaction Audit
- Access Audit
- System Audit

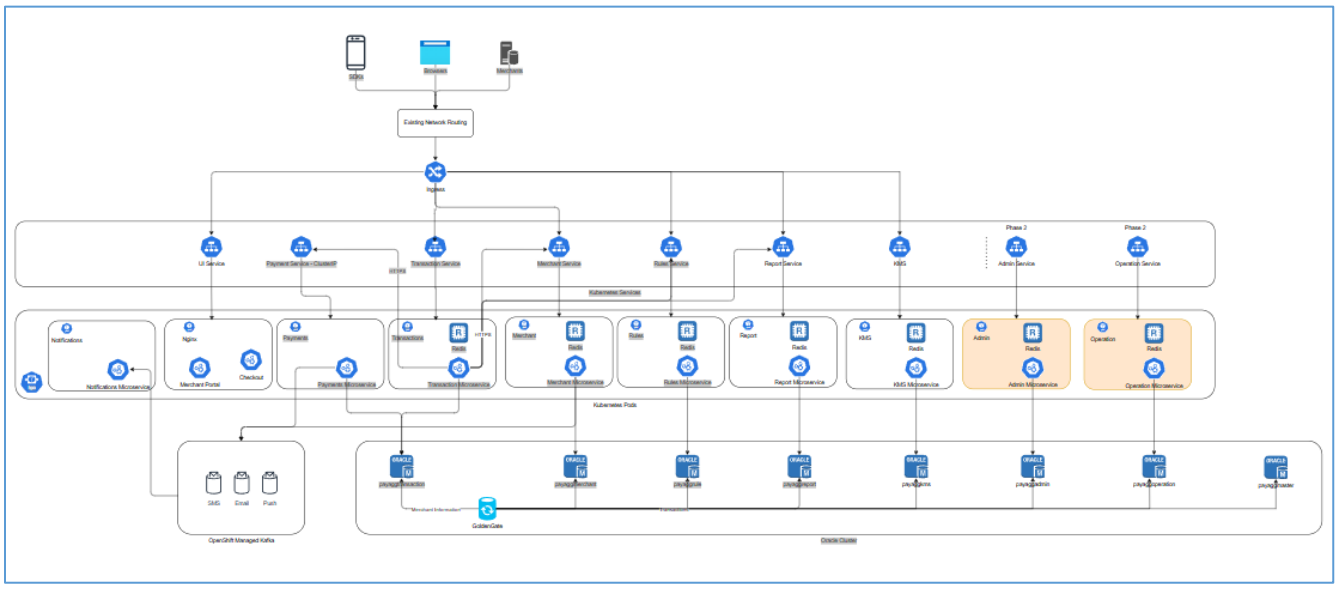
This Security API will interact with the Logs and Metrics Services during transaction processing.

The functionalities listed below will call the Audit Service API for its processing:

- Infra Logs
- Server Logs
- Application
- Telemetry

This Security API will interact with the Logs and Metrics Services during transaction processing.

SBI ePay 2.0 Architecture Diagram



There are three types of users:

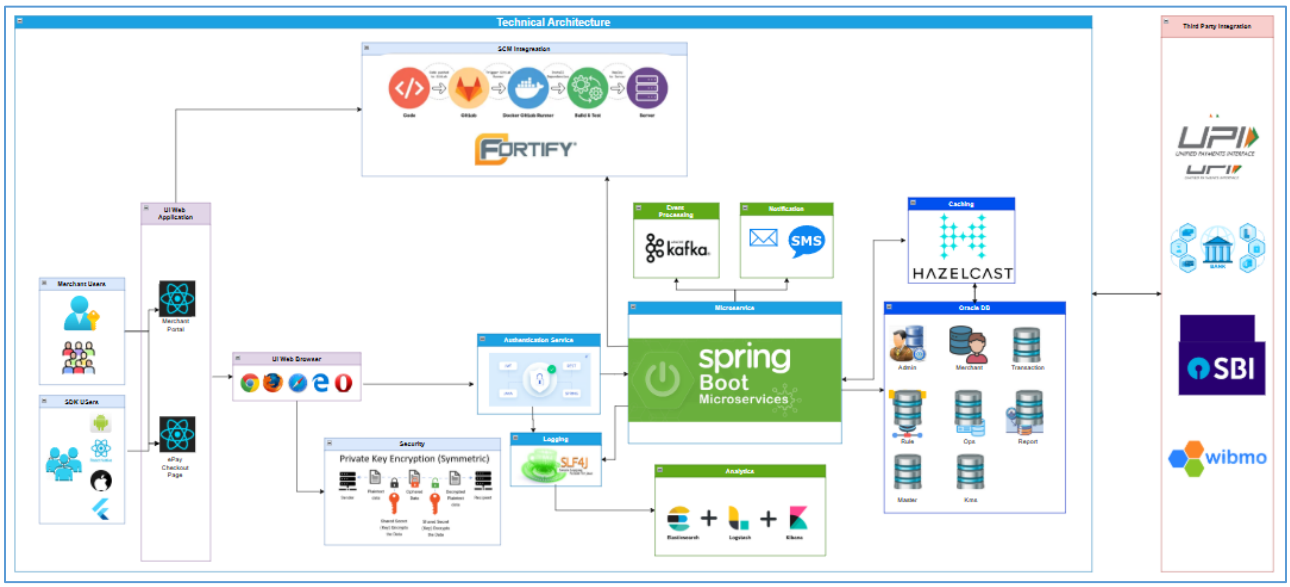
- Merchants
- Users using Browsers
- SDK Users

The above users interact with all the service APIs using Existing Network Routing Ingress.

- Merchant portal will interact with UI Service API, during which Notification Micro-service API will be called.
- Payment Micro-services will be used by Payment Service ClusterIP
- Transaction Micro-service, provided by Redis, will be used by Transaction Service
- Both the Payment and Transaction micro-services will be further interacting with the Oracle cluster - their respective databases, namely, payaggtransaction.
- Merchant Micro-services, provided by Redis, will be called by Merchant Service
- The Merchant Micro-service, will be using the Oracle cluster -its database payaggmerchant
- Payment Service and Merchant Service will interact with each other using HTTP protocol
- Rules Micro-services, again provided by Redis, will be called by Rules service
- The Rules Micro-service, will be using the Oracle cluster - its database payaggrule
- Report Micro-services, provided by Redis, will be used by Report Service. It will also call the Transaction Micro-service to get the reference of the transaction while generating any report.
- The Report Micro-service, will be using the Oracle cluster -its database payaggreport
- KMS Micro-services, provided by Redis, will be called by KMS
- The KMS Micro-service, will be using the Oracle cluster -its database payaggkms
- Phase 2 – Admin service will be utilizing the Admin Micro-services, provided by Redis, for which it will be fetching the data from Oracle cluster – payaggadmin

- Phase 2 – Operation service will be utilizing the Operation Micro-service, provided by Redis, which further will interact with the Oracle cluster – payaggoperation and payaggmaster to fetch the data.
- Finally, the entire data from the old database will be transferred to the new database using the gateway named GoldenGate.

llSBI ePay 2.0 Technical Architecture



Merchant and SDK users will access UI Merchant Portal and ePay Checkout page using UI Web Application.

UI Web Application supports the following web browsers such as:

- Google Chrome
- Mozilla Firefox
- Internet Explorer
- Opera

The above web browsers will continuously be interacting with Security services for authentication purpose, as and when required.

Encryption and Decryption of Data

- A data entered by sender, which is in the plain text format, will be encrypted using Shared Secret (Key)
- The above encrypted ciphered data will be decrypted back to the plain text format and the same will be shared to the recipient

Authentication services will further communicate with Logging provided by SLF4J, analytics by CLK and other Spring Boot Microservices.

Event processing by APACHE kafka, Email and SMS notification services will be used by the Spring Boot MicroServices.

Spring Boot Microservice will also use the Cache server by HAZELCAST and the Oracle DB for the following entities:

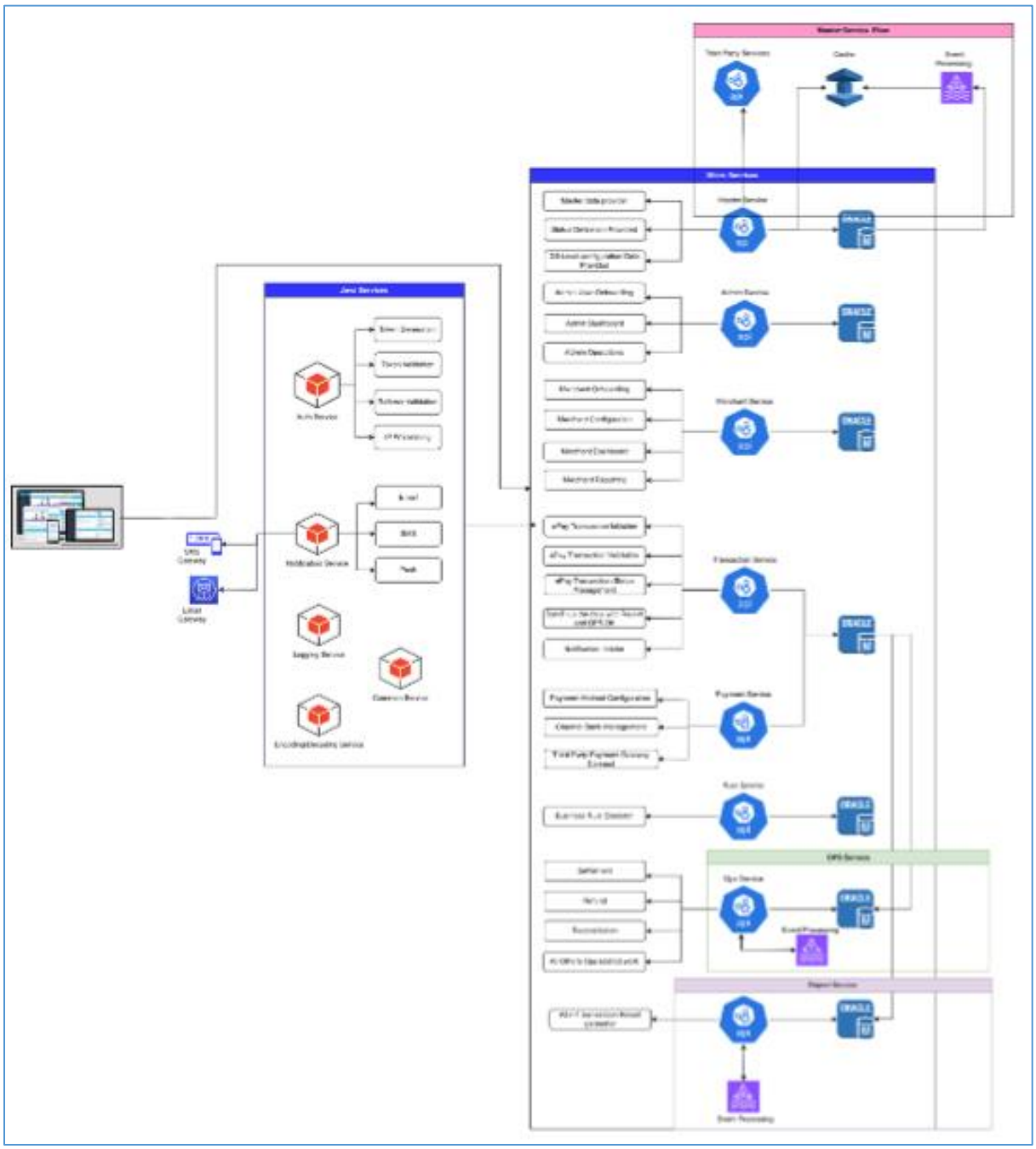
- Admin
- Merchant
- Transaction
- Rule
- Ops
- Report
- Master
- Kms

Spring Boot Microservice will use the SCM Integration provided by Fortify. Its internal processing is as follows:

- Code is pushed to GitLab server
- From GitLab runner, a trigger will be run to Docker GitLab Runner
- Internal build will be run along with application testing
- An application will further be deployed on the server

The entire Technical Architecture of SBIEPAY 2.0 will integrate itself with third-party services like UPI, wibmo, SBI, various other banks, and other financial institutes.

SBI ePay 2.0 Application Architecture



The above depicts as follows:

Whenever any transaction takes place, the following events occur in a particular sequence. During this execution, the various service APIs are called by their respective functions. The integration of all these services and its respective APIs help to build and integrate the architecture of the application.

Master Data Configuration and Update Activity

- Master Data Provider
- Status Definition Provided
- DB Level configuration Data Provided

Merchant Configuration

- Merchant Onboarding
- Merchant Configuration
- Merchant Dashboard
- Merchant Reporting

Admin Management

- Admin User Onboarding
- Admin Dashboard
- Admin Operations

Authorization service

- Token Generation
- Token Validation
- Referror Validation
- IP Whitelisting

Transaction Service

- ePay Transaction Initiation
- ePay Transaction Validation
- ePay Transaction Status Management
- Sync up the Data with Report and Ops DB
- Notification Initiator

Ops Service with Event Processing

- Settlement
- Refund
- Reconciliation
- All other Ops related work

Notification Service

- SMS Gateway
- Email Gateway

Common Service

Logging Service

Report Service

- All the Transaction-related Report Generation

Rule Service

Encoding/Decoding Service

All the above Service APIs will be fetching and storing the data to the Oracle DB