Horizon Payment Switch

Project Achitecture

Document Version: 1.0

Release Date: March 30, 2021

Document Change History				
No.	**Doc Version**	**Release Date**	**Author**	**Change Description**
1	1.0	30-03-2021	Mahyar Esteki	Create First Version

Contents

1 Abstract

Proper knowledge of the infrastructure and architecture of any system will lead to a better understanding of its performance. This information helps developers to reach a common view on the use and operation of the system. The purpose of presenting the architectural document of this project will be to create a common understanding of the structure and operation of the system. In this document you will get acquainted with system architecture, how services work, network architecture and more. This document contains general information and brief descriptions of the system architecture. Therefore, additional technical information will be provided in the relevant documents.

2 Project Schema

Project Horizon is a payment switch based on the ISO 8583 standard that manages its communications with other clients through RESTful APIs. Unlike previous examples, this switch stores some of the basic information related to its activities in a separate database. This method helps the payment switch to avoid the need for large references to the core banking system data for better control and execution of complex processes.

Horizon Project is a distributed payment switch that manages its operational division based on the province or state that sent the transaction. By using this method, the processing load due to the high number of banking transactions will be divided between several acquiree payment switches. This will ultimately increase the speed of response in banking operations. Horizon Payment Switch tries to avoid centralization of operations and processing load while fully synchronizing with the basic data in the core banking system.

The goal of Horizon project is the parallel implementation of several instances of this payment switch based on the number of specified zones. For each zone, an instance of the Horizon Payment Switch will run on a separate server. Through a reverse proxy or any load balancing tool, client requests can be sent to the server of the associated area based on the sender's IP address. The basis of this method is based on the **“Distributed Open Banking Theory”**, which is further explained in a separate document. This document is uploaded in the same folder on GitHub.

Data is distributed between zoned servers based on horizontal partitioning on financial transaction records. While basic data is centrally available to all servers. Through this method, the server of each zone will only have access to information related to its clients. However, it will be possible to view the records of all transactions according to the assigned access levels.

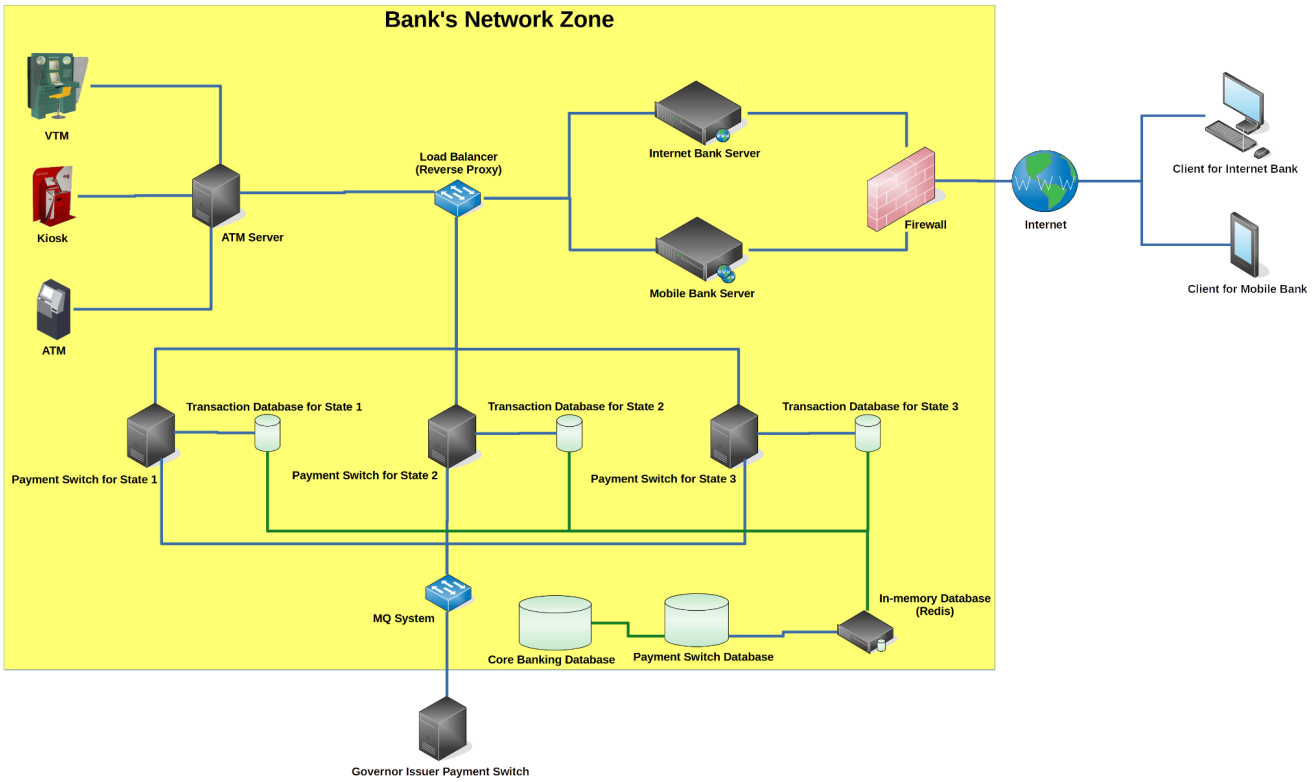

Figure 1: Project Network Diagram

3 System Architecture

Horizon Payment Switch acts as an open banking system and provides its services to clients in APIs. Therefore, it is necessary to use a multi-layer structure to separate the data layer from other parts. This structure even better organizes operations and their categories. In fact, in the horizon payment switch, an attempt has been made to provide a set of services to the clients through an OpenAPI channel, so that a client can provide suitable features for their main users without having to deal with the details and complexities of the services.

As a result, clients that communicate with the system will not have direct access to the data model, core services, and infrastructure functions. At first glance, the system architecture is very similar to MVC. However, there are two very obvious differences with this type of architecture. Horizon payment switch is a back system that does not have a view. In addition, the controller layer itself is divided into more precise sections. In fact, Horizon is a multi-layer system based on OpenAPI. The integration layer is the communication part of the system with the issuer switch that will reconstruct and send incoming messages according to the standards defined by the governing systems. This layer will be developed according to the needs of the target systems and its function will vary depending on the needs of the bank or financial institution using the Horizon payment switch.

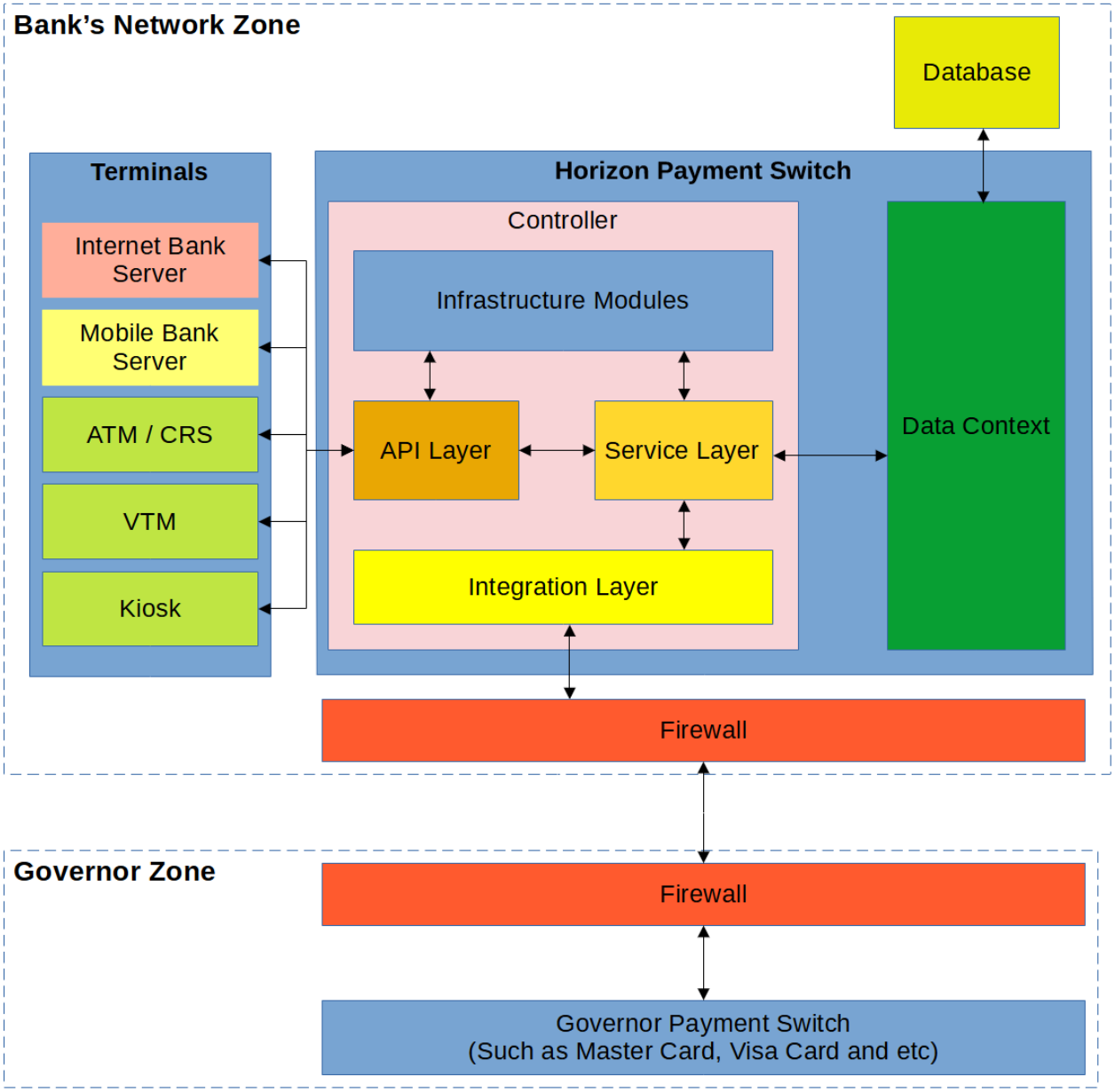

Figure 2: Project Architecture

4 Layers of the System

Horizon system consists of five main layers that determine the levels of operations, communication and data access. Each of these layers may be subdivided into a set of modules. But maintaining this overall structure is of particular importance for organizing a rapid and orderly development. These layers are:

- **API Layer:** This section is included a set of RESTful APIs which is used by terminals and the control panel. APIs call only certain services for the requested operation and have no independent access to the data model.
- **Service layer:** A set of services designed to perform activities, calculations and access to the data model are located in this section. The service layer is the only part of the system that has access to the data model.
- **Structure layer:** This part consists of a set of infrastructure modules that manage system operations such as log registration and encryption.
- **Data layer:** This section includes all data model entities.
- **Integration layer:** This section includes all the modules required to connect to the global banking network issuer switch. The main modules and functions of this layer must be designed and implemented by the user bank or financial institution.

5 Types of APIs

5.1 Business Types

Horizon Payment Switch uses four different ***business types*** of APIs, which are classified according to the type of activities performed. These four categories are:

- **ISO API:** These services include activities related to ISO 8583 compliant messages.
- **Non-ISO API:** These services will be customized by the developers of the owner bank or financial institution, which are also called value-added services (VAS).
- **User API:** Services and functions that each terminal uses to check basic information and switch settings.
- **Administrative API:** A set of functions used to edit the settings of a switch or any terminal connected to it.

5.2 Technical Types

Categorizing APIs based on technical indicators will significantly help to better understand the types of potential client requests. In general, there are two technical types of API in Horizon system, which are:

- **Normal APIs:** Used to request transactions or admin operations. In fact, they are the same APIs that are visible on many systems.
- **SSE APIs:** Used to implement **“server sent events”** model, which is used in cases such as receiving updates or the results of long-term operations.

6 Access Role Architecture

Depending on the type of messages sent by clients, there are three types of access roles in Horizon payment switch.

- **ISO requests authorization:** The system detects based on three main parameters such as the acceptor terminal ID (field number 41 in ISO 8583), the processing code (field number 3 in ISO 8583) and the MTI code: whether is acceptor terminal authorized to send such a this request or not?
- **Non-ISO requests authorization:** This item is used to check the access level when receiving requests related to value-added services. Based on acceptor terminal ID and used credentials, the system checks: whether is acceptor terminal authorized to send such a this request or not?
- **Administrative requests authorization:** This item is used to check the access level when receiving requests related to administration and configuration activities. Based on used credentials, the system checks: whether is the user authorized to send such a this request or not?

7 Access Role Levels

In the Horizon project, three access levels have been determined based on the requested activities of a terminal.

- **Access level of ISO messages:** This access level allows sending a message that complies with the ISO 8583 standard. This level of access is configurable between the message code processing process and the sender terminal role.
- **Access level for non-ISO messages:** This access level is used to check the user license of a specific terminal when sending messages related to value-added services.
- **Administrative Message Access Level:** This access level is used to verify the user license of a specific terminal when sending messages related to administrative services.

8 Horizontal Partitioning on Database

The basic information tables in the Horizon Payment Switch database must be synchronized with the same information in the core banking system database. The table structure in the Horizon Payment Switch database is divided into three sections: Basic Information, Settings, and Transaction Information. The first two types of tables will be recorded and maintained in the main database of the system. Transaction information will be stored in databases specific to each installed version of the horizon payment switch in a province or state in the target country as a horizontal partition of the main database.

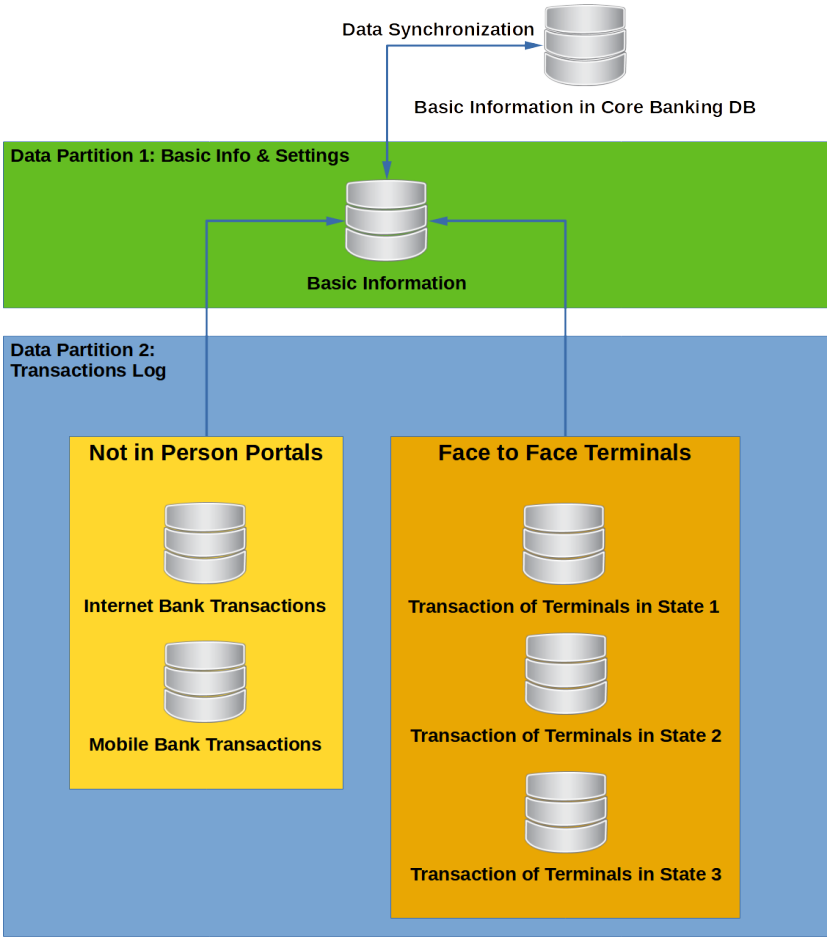

Figure 3: Database Partitioning Concept

9 Data Synchronization with Core Banking System

To speed up the validation process and reduce the number of requests to the core banking system, a set of basic data will be stored simultaneously in the Horizon Payment Switch database. This change in attitude towards the use of payment switches leads to a balanced distribution of processes in the banking system. This means that you no longer need to refer to the core banking database to check some basic information related to payment operations. But this data needs to be updated hourly or even instantly. For example, the exchange rate changes many times during the day. If the target bank terminals support multi-currency payments, it is vital to be aware of the exchange rate at all times.

Therefore, despite keeping this information in the payment switch databases, we need to update this information by the core banking database. To synchronize these two databases, we need to know what information and tables need to be updated. The following table lists the information that needs to be updated through the core banking system database.

No.	Data Title	Data Update Period
1	List of Counties	Yearly
2	List of States or Provinces	Yearly
3	List of Cities	Yearly
4	List of Currencies and Exchange Rates	Just in Time
5	List of Banks	Daily
6	List of Branches	Daily
7	List of Card Prefixes	Daily
8	List of Channel Types	Daily
9	List of Terminals	Daily

10 Platform

Programming Language	Python 3.8
API connection protocol	HTTP
API type	RESTful
API Technology	Flask (will be upgraded to FastAPI)
Database	PostgresSQL 12
In-memory Database	Redis
MQ System	Rabbit MQ

11 Entity Relationship Diagram (ERD)

The image below is for reference only. You can download this image in high quality through the relevant attachments on the GitHub website.