**Malayan Banking Berhad**



**MPI System Design Document**

**Retail & Corporate Internet Banking System for Maybank Philippines Incorporated**

26 October 2010

**Prepared By:**



Penril Datability (M) Sdn Bhd (816792-X)

Suite A-07-07 Plaza Mon’t Kiara

No. 2, Jalan Kiara, Mon’t Kiara

50480 Kuala Lumpur, Malaysia

Tel: (603) 6201 2622 Fax: (603) 6201 7622

**Document Administration**

**Document Amendment Log**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Version No**  | **Date Updated**  | **Description of Change**  | **Updated By** **<Name, Dept>**  | **Reviewed By** **<Name, Dept>**  | **Approved By <Name, Dept>**  |
| 1.0  | 20/10/2010  | Initial Release | Tan Lee Yong |  |  |

**Table of Contents**

[1 Document Purpose 4](#_Toc275882595)

[2 MPI Internet Banking System 4](#_Toc275882596)

[3 System Design 5](#_Toc275882597)

[3.1 System Components Overview 5](#_Toc275882598)

[3.2 System Design Overview 6](#_Toc275882599)

[3.3 Framework Design Overview 7](#_Toc275882600)

[4 JMS Management 8](#_Toc275882601)

[4.1 What is JMS 8](#_Toc275882602)

[4.2 JMS Platform 8](#_Toc275882603)

[4.3 MPI JMS Mechanism 8](#_Toc275882604)

[4.4 MPI JMS Message Design 8](#_Toc275882605)

[4.5 JMS Message Process Explanation 8](#_Toc275882606)

[5 Session Management 10](#_Toc275882607)

[5.1 What is Session 10](#_Toc275882608)

[5.2 Session Management 10](#_Toc275882609)

[5.3 Session Vulnerability 10](#_Toc275882610)

[5.4 Session Management Design Consideration 10](#_Toc275882611)

[5.5 Session Management Design 11](#_Toc275882612)

[6 Application Spec 12](#_Toc275882613)

**Table of Figures**

[Figure 1 MPI Internet Banking System 4](#_Toc275882770)

[Figure 2 M2U for MPI Component Diagram 5](#_Toc275882771)

[Figure 3 Application Architect Design Overview 6](#_Toc275882772)

[Figure 4 MVC Framework 7](#_Toc275882773)

[Figure 5 JMS Queue Process Diagram 9](#_Toc275882774)

[Figure 6 Session Management 11](#_Toc275882775)

# Document Purpose

This document contains a description of the Maybank Philippines Internet Banking System architecture. It provides a sub-system or component-level view of the application without going into significant detail about individual subsystems or components.

As application architecture, this document focuses primarily on the overall design of the application itself and the basic structure of the code organization. This document will be kept up-to-date throughout the life of this project to serve as a reference for all architecturally significant aspects of the application.

The more detailed technical specification will be provided in separate documents. The database design specification is provided in separate technical document.

# MPI Internet Banking System

The MPI Internet Banking System consists of 3 main applications, M2U Support System (MSS), MPI Retail Internet Banking (RIB) and MPI Business Internet Banking (BIB).

The MSS is the administration application designed for MPI Retail Internet Banking, MPI Business Internet Banking, MSS administration and Call Center usage. RIB is the application to serve MPI retail user and BIB is the application to serve MPI SME, corporate and business user to perform internet banking functionalities. All MPI applications are using the same authentication system (UPASS). Therefore, usernames are unique for all external and internal customers.

All the three applications are designed run on the J2EE Application server and develop on the MVC architecture pattern. Please see all the following sections on the details.

Figure MPI Internet Banking System

# System Design

## System Components Overview

The M2U system is staged with a 2 layers of service components, the Application Control Component and Service Component. The Application Control Component is the layer to control the system page flows and the integration for the page flow to the Service Component and the Integration Component. This layer is using the Model-View-Controller (MVC) software architecture pattern. The Service Component layer consist of different type of services to support the system like the control of page Navigation, authentication component (UPASS), the Access Matrix component, etc.. The Integration Component layer consists of the Java Message Service (JMS) component which shall integration with the MPI Host System.

Figure M2U for MPI Component Diagram

## System Design Overview

The below diagram is the overview of the MPI M2U Internet Banking application architecture framework. As shown in the diagram, the Web Client is implements using Struts framework. A Business Functions layer, on the Application Server, consists of business logics and acts as the communication layer to other services. Web Client only communicates with the Business Functions layer and is transparent to other services.

Figure Application Architect Design Overview

## Framework Design Overview

Figure 3 depicts the architecture of MVC system and the design to decouple different layers at the Business layer.

All Action classes will have methods to call from DAO zone. These DAO will have the real code to connect to the external systems like Database, CICS, RSA, etc. Optionally, all connection objects shall be received from centralized Class, so that any new connection objects or modifications shall be controlled from one class. All Data returned from DAO are in normal Java Bean Objects, not any system specific object. This will help in easy integration with any system in future stages.



Figure MVC Framework

# JMS Management

## What is JMS

The Java Message Service (JMS) API is a Java Message Oriented Middleware API (MOM) for sending messages between two or more clients. JMS is a part of the Java Platform, Enterprise Edition, and is defined by a specification developed under the Java Community Process as JSR 914. It is a messaging standard that allows application components based on the Java 2 Platform, Enterprise Edition (J2EE) to create, send, receive, and read messages. It allows the communication between different components of a distributed application to be loosely coupled, reliable, and asynchronous.

## JMS Platform

Implementing JMS is platform specific configuration. For this MPI project, the JMS design is based on Oracle WebLogic 10.

## MPI JMS Mechanism

Figure 4 shows the overview design of the JMS component. The JMS component consists of 2 subcomponents, JMS Client (also called Producer) and JMS Server (Consumer / Producer).

JMS Client will receive a request, drop the request message to the queue, wait for the response message and return the response message once obtained before timeout. The queue request is demonstrated in Figure 4 start from step 1.1 and completed of receiving response at step 1.16.

JMS Server will read a message from queue, process it, get the response info, construct the response message and drop the response message to the queue. The process message mechanism is demonstrated in Figure 4 start from step 2.1 and completed of dropping response message at step 2.4.

## MPI JMS Message Design

Each JMS message is identified by a **unique message id**. A unique id shall be generated and assigned to the Enterprise Java Bean (EJB) at the JMS Client stage. This id will be used for the entire lifecycle of the queue process. At the JMS Server stage, once a message is being processes, the same message id shall be used to construct the response message before drop it to the queue.

## JMS Message Process Explanation

The diagram shows that the service layer will drop a request message to the JMS Response Queue and then wait at the JMS Response Queue for the response. Once the response message is received, the message will be processed at the M2U Service Layer.

At the JMS Server layer (Process layer), the Request MDB will pick up a message from the JMS Request Queue, convert it to host message format and drop the message to CICS through socket. The Request MCB will get the response from CICS and then drop the message to the JMS Response Queue to complete the task.



Figure JMS Queue Process Diagram

# Session Management

## What is Session

A Session is semi-permanent interactive information between a computer and user. Sessions are used to compensate with the stateless condition of the HTTP protocol and allows storage of information that is associated with the client for the duration of the client’s visit. There is a unique identification string associated to a session called as Session ID (SID).

## Session Management

Session management is the technique used by the web developer to make the stateless HTTP protocol support session state.

## Session Vulnerability

* Attacks focus on retrieving a valid session key.
* Stealing session ID allows malicious user to assume permissions of legitimate user
* Session attacks consist of two major categories:
* Session hijacking
* Session fixation

## Session Management Design Consideration

No cookies shall be use in this MPI project. Below are the points taken as the consideration to ensure good session management and reduce the vulnerability:-

* Use of Strong Encryption on all Transmissions
* Store only Session ID on Client side
* Perform Sanity Checks to Detect Session Hijacking
* Expire session after Inactivity
* Do not make Session IDs Viewable
* Select Good Session Identifier
* Prevent Cross-Site Scripting (XSS) Vulnerabilities
* Force Server-side Session ID creation
* Double Check Critical Operations
* Provide Secure Logout
* Securely Store the Server side session map
* Expire the pages (to Prevent Caching)
* Make the Session ID Dynamic with Hijack Attempt Detection
* Require Re-Authentication after Maximum Login Limit

## Session Management Design

The session management design has taken all the consideration in section 5.4. The design it to keep the session short life and prevent reuse of the session.

##



Figure Session Management

# Application Spec

|  |  |
| --- | --- |
| Item |  |
| Web Server | Sun Web Server |
| Application Server | Oracle WebLogic 10.3.1 |
| Database | Oracle 10g |
| Operating System | Sun Solaris |
| Use of programming language | Java, JavaScipt, JSP, AJAX, HTML, XML,  |
| Use of Framework | Struts 2.0, Hibernate 3, JMS, Web Service |
| Platform | JDK 1.6 |
| Development Tool | Eclipse |
| Source / Document Repository | Mercurial / Sub Version |