Distributed Polling System
Design Specification

Version 1.0
# Table of Contents

1. Introduction 3  
   1.1 Purpose of this document 3  
   1.2 Intended Audience 3  
   1.3 Scope 3  
   1.4 Definitions and acronyms 3  
      1.4.1 Definitions 3  
      1.4.2 Acronyms and abbreviations 4  
   1.5 References 4  

2. External interfaces 4  

3. Software architecture 4  
   3.1 Guiding principles in defining architecture 4  
   3.2 How the DPS architecture originates from above guidelines 4  
   3.3 Connectivity choices for applications 5  
      3.3.1 Web-DPS Application: 5  
      3.3.2 SMS Gateway 6  
      3.3.3 Email Server: 7  
   3.4 Conceptual design 7  
      3.4.1 System level conceptual design 7  
      3.4.2 System Interface Diagram 9  
      3.4.3 WEB-DPS application design 11  
   3.1. System specification 12  
   3.2. Error handling 12  

4. Detailed software design 12  

5. Approvals 18
1. Introduction

1.1 Purpose of this document
This document is a deliverable of architecture and design phase from DPS project team as derived from requirement analysis. The main purpose of this document is to emphasize detailed overview of architecture, design and will be a reference during implementation phase. The document will be fruitful to provide a clear understanding of the complex DPS and design each module.

1.2 Intended Audience
Intended Audience of this document is:
- Customers
- Supervisors
- Project Team
- Other interested parties

1.3 Scope
This document will give detailed design description of each component of DPS. Following applications and their integration are in the scope of this design document:
- Design of core application
- SMS Gateway
- Email server
- Web application

1.4 Definitions and acronyms

1.4.1 Definitions

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Definitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business Process</td>
<td>A set of business activities executed either in sequence or in parallel to accomplish business functionality.</td>
</tr>
<tr>
<td>Service</td>
<td>A service (consists of rules) that takes some predefined input and produces a specific output.</td>
</tr>
<tr>
<td>Interface</td>
<td>An interface is a communication between two systems or layers to perform a specific task.</td>
</tr>
<tr>
<td>Polling</td>
<td>Monitoring a specific software component with a predefined time interval.</td>
</tr>
<tr>
<td>SMS Gateway</td>
<td>A gateway for communicating telecom software applications with SMSC. SMSC ultimately interacts with mobile devices.</td>
</tr>
<tr>
<td>Adapters</td>
<td>A software component that enables communication between two heterogeneous applications.</td>
</tr>
<tr>
<td>System</td>
<td>A system consists of interdependent applications or components that interact with each other and produces complex business functionalities.</td>
</tr>
<tr>
<td>Application</td>
<td>An application is a subset or subsidiary of a complex system</td>
</tr>
</tbody>
</table>
1.4.2 Acronyms and abbreviations

<table>
<thead>
<tr>
<th>Acronym or abbreviation</th>
<th>Definitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPS</td>
<td>Distributed Polling System</td>
</tr>
<tr>
<td>MSISDN</td>
<td>Mobile Station International Subscriber Directory Number</td>
</tr>
<tr>
<td>IMSI</td>
<td>International Mobile Subscriber Identity</td>
</tr>
<tr>
<td>ESB</td>
<td>Enterprise Service Bus</td>
</tr>
<tr>
<td>SMSC</td>
<td>SMS Center</td>
</tr>
</tbody>
</table>

1.5 References

<table>
<thead>
<tr>
<th>No.</th>
<th>Document Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Requirements Specification document</td>
<td>Requirements specification for DPS</td>
</tr>
</tbody>
</table>

2. External interfaces

The system itself is exposed to the user through three different ways. Main application GUI has system functionality to the system administrator and the members who are interested in creating a poll for decision making. SMS and Email application interface are available for the members of a poll to receive and respond to the decision making.

3. Software architecture

DPS architecture has been modeled considering scalability, loose coupling, and user friendly interaction and so on. High level system architecture diagram of DPS is depicted below.

3.1 Guiding principles in defining architecture

Following guiding principles were considered during system architecture finalization.

- There is a business need to exchange the data from one system to another system in real-time.
- Need for message transformation (change of data format or combining of multiple messages into a single message or so) and routing
- Need for the systems to be loosely coupled through asynchronous messaging.
- Re-usability of the interface in that the same data needs to be sent from one system to multiple systems
- Process integration (Process orchestration) wherein a business transaction spans over multiple systems and message need to be processed from one system to another
- Need of guaranteed message delivery
- Need of security in application interactions for business purposes
- Smooth and effective error handling mechanism

3.2 How the DPS architecture originates from above guidelines

During designing the architecture we have considered DPS as a software system which consists of three software applications

- SMS Gateway
- web-DPS application and
- Email server.

For scalability, loose-coupling and technology independence we have introduced middleware-the core of our system that takes care of message transformation, routing, hides the complexity and interaction among the above depicted three software applications. Middleware provides hub and spoke architecture (Figure 3-1: System Architecture) for the system where middleware is at the hub and software applications are at the perimeter of bicycle-wheel linked with hub through spoke called ‘adapters’ in software terminology.

Software applications do not need to bother how the messages should be exchanged among
themselves. Rather it’s the middleware who takes care of low level technical details and communicates with each application using different technologies. As middleware communicates in different manner with different applications, applications are built in any platform and technology.

![Diagram of System Architecture](image)

**Figure 3-1: System Architecture**

Following is the layered architecture of the custom built WEB-DPS application as depicted in system architecture diagram (Figure 3-1: System Architecture).

![Diagram of Layered Architecture Stack](image)

**Figure 3-2: Layered Architecture Stack of Web-DPS Application**

3.3 Connectivity choices for applications

3.3.1 Web-DPS Application:

Four different approaches were considered for integration of Web-DPS application with EAI layer.

1. Integration using XML/HTTP or HTTPS
Technologies used in ‘business logic and integration layer’ of Web-DPS application allows to invoke an exposed service at EAI layer using XML message over HTTP/HTTPS protocol. (Web-DPS ApplicationÆEAI Layer)

Similarly, from EAI layer also, exposed service at WEB-DPS application can be invoked using XML message over HTTP/HTTPS protocol (EAI LayerÆWEB-DPS Application)

2. Integration using Web-Service

• An EAI layer service can be exposed as web-service and can be invoked from ‘Business logic and integration layer’ of WEB-DPS application as web-service (WEB-DPS ApplicationÆEAI Layer)

• Web-services can be implemented at ‘business logic and integration layer’ of WEB-DPS application as business services or work-flow and the same can consume the WSDL document through HTTP transport (EAI LayerÆWEB-DPS Application)

3. JMS Approach

As WEB-DPS application will be deployed on J2EE compliant application server, it is quite possible to trigger business messages in both ways over JMS queues.

JMS listener plug-in of Jboss ESB allows listening over JMS queues and triggering ESB services using those messages.

4. JDBC Adapter approach

JDBC listeners can be used to poll a configured event table at WEB-DPS application end and the listener thread can trigger Jboss ESB service with an event populated in the event table.

5. Stored Procedure approach

Stored procedure can be implemented as business service at Web-DPS application database layer and can be invoked from middleware layer for synchronous communication.

Comparative study of Connectivity Choices

<table>
<thead>
<tr>
<th>Feature</th>
<th>XML/HTTP or HTTPS</th>
<th>Web-Service</th>
<th>JMS</th>
<th>JDBC Adapter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synchronous communication</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Asynchronous communication</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Supports guaranteed delivery</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Type of coupling between WEB-DPS application and EAI layer</td>
<td>Loose</td>
<td>Loose</td>
<td>Loose</td>
<td>Loose</td>
</tr>
<tr>
<td>Development Time</td>
<td>High</td>
<td>High</td>
<td>High</td>
<td>Less</td>
</tr>
</tbody>
</table>

Adopted Approach

WEB-DPS Application Outbound:

By considering all aspects, we have finalized JDBC adapter approach for WEB-DPS application outbound communication where WEB-DPS application will trigger an event to a custom event table located in WEB-DPS application database.

An EAI listener thread will poll this table and will pick up an event as soon as it is inserted in this table. The event is then dispatched to a subscribed service at EAI layer.

WEB-DPS Application Inbound:

Stored Procedure API at Web-DPS application database will be used for synchronous communication requirement from EAI layer to WEB-DPS application. EAI layer expects a response from WEB-DPS application, once EAI layer invokes the Web-DPS business service.

3.3.2 SMS Gateway

• During sending poll details to members, project requirement suggests receiving an acknowledgement from SMS gateway to track whether a member is reachable in his mobile or not.

To fulfill the same, we have decided to invoke the service built at SMS gateway end, synchronously
from EAI layer to receive acknowledgement in the same thread. (EAI Layer → SMS Gateway)

- Project requirement suggests receiving SMS responses. But there is no synchronous requirement.
  SMS gateway can collect SMS responses from members and put in a custom event table configured in its database. (SMS Gateway → EAI Layer)

We analyzed to adopt JDBC adapter approach where a listener thread can poll the event table, pick up events from that table and send to EAI layer for further processing.

**Adopted Approach**

**SMS Gateway Outbound:**
- JDBC Adapter approach

**SMS Gateway Inbound:**
- Synchronous communication, invoke SMS gateway service using XML over HTTP/HTTPS or web-service or stored procedure.
  Any one of these approach will be finalized during implementation.

### 3.3.3 Email Server:

Project requirement suggest to intimate poll details to members through email and to collect the responses.

- Sending email to members will be handled using services built at EAI layer to communicate with Email server (EAI Layer → Email server)
- Receiving email from email server will be handled using Email adapter that can poll an inbox of an email account (for example, middleware@middleware.com) (Email server → EAI Layer)

**Adopted Approach**

**Email Outbound:**
- Email adapter to poll inbox of an email account

**SMS Gateway Inbound:**
- Using custom built service at EAI layer.

**Future considerations:**

- *As per the requirement, in future, if any other mode of communication with the members to be incorporated (for example voice mail), we need to add one more application to the ESB and depicted in architecture diagram (Figure 3-1: System Architecture).*
  - *No need to change whole system. It’s only a plug-in to the existing system.*

### 3.4 Conceptual design

In this section we will depict both **system level** and **core application** level (WEB-DPS application) design details.

- System level conceptual design (using sequence diagrams)
- Interface requirement design
- WEB-DPS application design (using class diagram)

#### 3.4.1 System level conceptual design

At high level, Distributed polling system requires three **business processes** as delineated below.

1. Intimate Poll Details
2. Capture SMS Response
3. Capture Email Response
**Intimate Poll Details**

As depicted in the below sequence diagram, Poll created by WEB-DPS application will be picked up by listener thread and will be intimated to members through SMS and email using the mechanism of interaction as discussed in section-3.

**Figure 3-3: Intimate Poll Details**
Following UML sequence diagram depicts (in this business process) how member’s poll responses will be captured by SMS gateway and the same will be updated at WEB-DPS application.

![Sequence Diagram](image)

**Figure 3-4: Capture SMS response**

**Capture Email Response**
Below sequence diagram depicts the business process how email response is captured in WEB-DPS application.

![Sequence Diagram](image)

**Figure 3-5: Capture Email response**

3.4.2 **System Interface Diagram**
Following diagram depicts DPS interfacing requirements among different applications and their existence in the whole system view or perspective. Interestingly, it also delineates single and parallel steps (in different legend) that can occur or can co-exist in the system’s evolution.
At high level, Distributed polling system requires three business processes as mentioned in section-3.1.1.
1. Intimate Poll Details
2. Capture SMS Response
3. Capture Email Response
We will specify here the interfaces involved in each business process according to the above diagram (Figure 3-6: System Interface Details).

**Interface naming convention.**
INT_BPI_01: INT for interface
_BP1 for business process 1
_01 for interface number 01

**Intimate Poll Details:**
As per the diagram (Figure 3-6: System Interface Details),

INT_BPI_01 depicts Step-1
INT_BPI_02 depicts Step-2
INT_BPI_03 depicts Step-3
INT_BPI_04 depicts Step-4

**Capture SMS Response**
As per the diagram (Figure 3-6: System Interface Details),

INT_BP2_01 depicts Step-5 (Receive member’s SMS response)
INT_BP2_02 depicts Step-6 (Archive the event received by middleware for tracking purposes)
INT_BP2_03 depicts Step-6 (Invoke service to update voting response)

**Capture Email Response**
As per the diagram (Figure 3-6: System Interface Details),
INT_BP3_01 depicts Step-5 (Poll response email from server)
3.4.3 **WEB-DPS application design**

Following section describes the UML class diagram for WEB-DPS application as defined in architecture diagram (Figure 3-2: Layered Architecture Stack of Web-DPS Application)

![UML Class Diagram of Web-DPS Application](image)

**Figure 3-7: UML Class Diagram of Web-DPS Application**

INT_BP3_02 depicts Step-6 (Invoke service to update voting response)
3.1. System specification
Due to versatile nature of the architecture, scalability and for loose coupling design, a lot of software components need to interact with each other.
Following is the list of software components to be used for DPS implementation.

<table>
<thead>
<tr>
<th>Sr no</th>
<th>Software Component</th>
<th>Product Suite</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Jboss ESB</td>
<td>jbossesb-server</td>
<td>4.4</td>
</tr>
<tr>
<td>2.</td>
<td>Jboss Application Server</td>
<td>jboss</td>
<td>4.2</td>
</tr>
<tr>
<td>3.</td>
<td>Apache Ant</td>
<td>Apache</td>
<td>1.7.1</td>
</tr>
<tr>
<td>4.</td>
<td>JDBC Listener</td>
<td>jbossesb-server</td>
<td>4.4</td>
</tr>
<tr>
<td>5.</td>
<td>Email Listener</td>
<td>jbossesb-server</td>
<td>4.4</td>
</tr>
<tr>
<td>6.</td>
<td>MySQL DB</td>
<td>MySql</td>
<td>5.0</td>
</tr>
<tr>
<td>7.</td>
<td>Email Server</td>
<td>Winmail</td>
<td>4.6</td>
</tr>
<tr>
<td>8.</td>
<td>Eudora</td>
<td>Eudora web browser</td>
<td>7.1</td>
</tr>
</tbody>
</table>

3.2. Error handling
One of the major aspects of a sustainable software system is its proper handling of faulty scenarios. DPS will handle maximum of its error handling scenarios at EAI layer. In this project we have both the following type of interfaces
- Synchronous interfaces
- Asynchronous interfaces
Treatment for both these type of interfaces will be different and according to the business needs. We will produce a separate error handling document for this and will embed here for clear understanding.

4. Detailed software design
As discussed above, DPS has three main business processes as below:
1. Intimate Poll Details
2. Capture SMS Response
3. Capture Email Response
Using service oriented architecture, we will build units of services at EAI layer those exist individually and as a whole fulfills the process requirement.

**Intimate Poll Details**
This process intimates members on poll details through SMS and Email based upon business logic applicable.

*Involved Applications*
- WEB-DPS Application
- SMS Gateway
- Email server

*Initiating Application*
- WEB-DPS Application *(Pull Mechanism from EAI)*

*Interface Matrix*
### Interface ID

<table>
<thead>
<tr>
<th>Interface ID</th>
<th>Source Application</th>
<th>Target Application</th>
<th>Protocol</th>
<th>Volume/Frequency</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>INT_BP1_01</td>
<td>WEB-DPS Application</td>
<td>EAI</td>
<td></td>
<td></td>
<td>Medium</td>
</tr>
<tr>
<td>INT_BP1_02</td>
<td>EAI</td>
<td>WEB-DPS Application</td>
<td></td>
<td></td>
<td>Medium</td>
</tr>
<tr>
<td>INT_BP1_03</td>
<td>EAI</td>
<td>SMS Gateway</td>
<td></td>
<td></td>
<td>Complex</td>
</tr>
<tr>
<td>INT_BP1_04</td>
<td>EAI</td>
<td>Email Server</td>
<td></td>
<td></td>
<td>Complex</td>
</tr>
</tbody>
</table>

**Service Diagram**

Following diagram depicts the units of services to be built for the business process.

![Service Diagram](image)

**Figure 4-1: Poll intimation EAI services**

**Service Steps**

<table>
<thead>
<tr>
<th>EAI Layer Step</th>
<th>Description</th>
<th>Target Application</th>
<th>EAI Layer Service</th>
<th>Target application service</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. SMS sending request</td>
<td>This service sends a SMS to a recipient mobile station (mobile device)</td>
<td>SMSGateway</td>
<td>SMSinteraction</td>
<td>sendSMS</td>
</tr>
<tr>
<td>2. Email sending request</td>
<td>Sends an email to a target email-recipient</td>
<td>Email server</td>
<td>sendEmail</td>
<td>NA</td>
</tr>
<tr>
<td>3. Status update in WEB-DPS application</td>
<td>SMS and Email sending status is updated back at WEB-DPS application</td>
<td>WEB-DPS application</td>
<td>updateTable</td>
<td>DB update</td>
</tr>
</tbody>
</table>
Use Case

Goal
Intimating members with poll details through SMS and email

Involved Applications
SMS Gateway, Email server, WEB-DPS Application

Triggering Event
As soon as an event is triggered in the Middleware_Event_Table located at WEB-DPS application database

Entry Point
Once a poll is created at WEB-DPS application

Exit Point
Successful sending of SMS, EMAIL and status update at Middleware_Archive_Table

Process Flow
As soon as an event is inserted in Middleware_Event_Table, a JDBC listener that is always polling this table picks up the event and send it to integration ESB (enterprise service bus) where a service is subscribed to that event.

Success Terms
When all the steps mentioned in the Business service flow diagram are successfully executed.

Failure Terms
Please refer to DPS_Error_Handling_strategy

Business Rules

<table>
<thead>
<tr>
<th>Sr No</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Verify the ‘Severity’ parameter from the source message and based upon we need to send either SMS or Email or both</td>
</tr>
<tr>
<td>2.</td>
<td>Verify ‘isPDF’ parameter value to decide whether PDF document to be generated or not</td>
</tr>
<tr>
<td>3.</td>
<td>Consider the ‘subscriber’ parameter values for sending email and SMS to the specified members only</td>
</tr>
<tr>
<td>4.</td>
<td>Use ‘isAnonymous’ parameter value to intimate members whether they can vote anonymously or not</td>
</tr>
</tbody>
</table>

Capture SMS Response
This process captures the SMS voting response from members and updates at WEB-DPS application for calculation.

Involved Applications
- WEB-DPS Application
- SMS Gateway

Initiating Application
- SMS Gateway (Pull Mechanism from EAI)

Interface Matrix

<table>
<thead>
<tr>
<th>Interface ID</th>
<th>Source</th>
<th>Target</th>
<th>Protocol</th>
<th>Volume/Frequency</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>INT_BP2_01</td>
<td>SMS Gateway</td>
<td>EAI</td>
<td></td>
<td>Medium</td>
<td></td>
</tr>
<tr>
<td>INT_BP2_02</td>
<td>EAI</td>
<td>SMS Gateway</td>
<td></td>
<td>Medium</td>
<td></td>
</tr>
<tr>
<td>INT_BP2_03</td>
<td>EAI</td>
<td>WEB-DPS Application</td>
<td></td>
<td>Complex</td>
<td></td>
</tr>
</tbody>
</table>

Service Diagram
Following diagram depicts the units of services to be built for the business process.
Service Steps

<table>
<thead>
<tr>
<th>SI No</th>
<th>EAI Layer Step</th>
<th>Description</th>
<th>Target Application</th>
<th>EAI Layer Service</th>
<th>Target application service</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Archive SMS response</td>
<td>This service captures SMS vote from SMS Gateway and update status at SMS gateway database</td>
<td>SMSGateway</td>
<td>captureSMS</td>
<td>NA</td>
</tr>
<tr>
<td>2.</td>
<td>Update SMS response details</td>
<td>Status of This service sends voting response details to WEB-DPS application</td>
<td>WEB-DPS Application</td>
<td>updateVotingResponse</td>
<td>captureVoting</td>
</tr>
</tbody>
</table>

Use Case

<table>
<thead>
<tr>
<th>Goal</th>
<th>To capture SMS voting and to update in WEB-DPS application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Involved applications</td>
<td>SMS Gateway, WEB-DPS Application</td>
</tr>
<tr>
<td>Triggering Event</td>
<td>As soon as an event is triggered in the SMSGateway_EVENT_TABLE located at SMS Gateway application database</td>
</tr>
<tr>
<td>Entry Point</td>
<td>Once a subscriber responds or votes through SMS</td>
</tr>
<tr>
<td>Exit Point</td>
<td>Successful update of SMS voting response at WEB-DPS application</td>
</tr>
<tr>
<td>Process Flow</td>
<td>As soon as an event is inserted in SMSGateway_EVENT_TABLE, a JDBC listener that is always polling this table picks up the event and send it to integration ESB where ‘captureSMS’ is subscribed to that event. Using ‘captureSMSVoting’ service, finally the response is updated at WEB-DPS application</td>
</tr>
<tr>
<td>Success Term</td>
<td>When all the steps mentioned in the Business service flow diagram are successfully executed.</td>
</tr>
<tr>
<td>Failure Term</td>
<td>Please refer to DPS_Error_Handling_strategy</td>
</tr>
</tbody>
</table>

Business Logic

<table>
<thead>
<tr>
<th>Sr No</th>
<th>Business Logic</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Extract the required ‘Topic-id’ or the unique string for updating the response in WEB-DPS application using these parameters as key values</td>
</tr>
</tbody>
</table>
2. Populate the MSISDN and IMSI number both to the WEB-DPS application to verify that the response is from right member and not fraud activity.

**Capture Email Response**

This process captures the Email voting response from members and updates at WEB-DPS application for calculation.

**Involved Applications**

- WEB-DPS Application
- Email server

**Initiating Application**

- Email Server (Pull Mechanism)

**Interface Matrix**

<table>
<thead>
<tr>
<th>Interface ID</th>
<th>Source</th>
<th>Target</th>
<th>Protocol</th>
<th>Volume/Frequency</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>INT_BP3_01</td>
<td>Email Server</td>
<td>EAI</td>
<td></td>
<td></td>
<td>Medium</td>
</tr>
<tr>
<td>INT_BP3_02</td>
<td>EAI</td>
<td>WEB-DPS Application</td>
<td></td>
<td></td>
<td>Complex</td>
</tr>
</tbody>
</table>

**Service Diagram**

Following diagram depicts the units of services to be built for the business process.

![Service Diagram](image)

**Service Steps**

<table>
<thead>
<tr>
<th>Sl No</th>
<th>EAI Layer Step</th>
<th>Description</th>
<th>Target Application</th>
<th>EAI Layer Service</th>
<th>Target application service</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Update email response details</td>
<td>Status of Email voting response is updated at WEB-DPS application</td>
<td>WEB-DPS Application</td>
<td>updateVotingResponse</td>
<td>captureVoting</td>
</tr>
</tbody>
</table>

**Use Case**

<table>
<thead>
<tr>
<th>Goal</th>
<th>To capture email voting and to update in WEB-DPS application</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Involved applications</strong></td>
<td>Email server, WEB-DPS Application</td>
</tr>
<tr>
<td><strong>Triggering Event</strong></td>
<td>As and when any new email arrives at the response email account, email adapter polls that email and sends to subscribed service at EAI layer.</td>
</tr>
<tr>
<td><strong>Entry Point</strong></td>
<td>Once a subscriber responds or votes through EMAIL</td>
</tr>
<tr>
<td><strong>Exit Point</strong></td>
<td>Successful update of EMAIL voting response at WEB-DPS application</td>
</tr>
</tbody>
</table>
Email adapter continuously monitors the inbox folder of response email account and as soon as it gets any new email, the same is captured by EAI service built at EAI layer. This service invokes WEB-DPS application exposed service to update the response as vote at WEB-DPS application.

Success Term
When all the steps mentioned in the Business service flow diagram are successfully executed.

Failure Term
Please refer to DPS_Error_Handling_strategy

**Business Logic**

<table>
<thead>
<tr>
<th>Sr No</th>
<th>Business Logic</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Extract ‘FromAddress’ and the unique key for example ‘Topic-id’ to update at WEB-DPS application</td>
</tr>
</tbody>
</table>

**WEB-DPS Application Database Design**
Following is the logical design of our database, required for implementing WEB-DPS application.

![Logical design of WEB-DPS application database](image)

Figure 4-4: Logical design of WEB-DPS application database
Entity Relationship Diagram of WEB-DPS application database

Following diagram depicts the entity relationship diagram of WEB-DPS application database at entity level.

Figure 4-5: Entity Relationship Diagram

5. Approvals

<table>
<thead>
<tr>
<th>Name</th>
<th>Title</th>
<th>Date yyyy-mm-dd</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mr. Miguel Felder</td>
<td>Project Customer</td>
<td>TBD</td>
<td></td>
</tr>
<tr>
<td>Mr Rikard Land</td>
<td>Project Supervisor</td>
<td>TBD</td>
<td></td>
</tr>
</tbody>
</table>