# Employee Management Service

## Technical Documentation

**Version:** 8.0  
**Author:** Prashant Bhaskar Gharat  
**Last Updated:** February 20, 2025

## Table of Contents

1. [System Overview](#Xfc21e1f09b6b3d7a2b0a47e7b5773e928698810)
2. [Architecture](#X136a724f0ad6bedecc70d52a088da11b9355002)
3. [Component Specifications](#Xf07d9816c5b987043e5be4f193d4ea62a23ce35)
4. [Database Design](#Xe7d51658d8f2379c0969f4580b5bec6ca5fc228)
5. [Process Flows](#Xbfaf11a5cba2f3fc9c099abce2d4feb828093d7)
6. [Error Handling](#X263082721bea3ec65046e5d9ecdeecae35f702b)
7. [Configuration](#X2e0c9fd994380ccbac697d5907fa9066d612ab5)
8. [Deployment](#X7b7a6f2d5c835aacb9671ab9743ccc215f168f5)
9. [Monitoring and Operations](#Xc7afd063effa26a79e2564c1386f72fb4aa2ed5)
10. [Troubleshooting Guide](#X1323b945d6cc2cfdc2bd668c1d6cb8088fffc6f)

## 1. System Overview

### 1.1 Purpose

The Employee Management Service is a TIBCO BusinessWorks 5.x application designed to automate employee lifecycle operations, including recruitment, promotion, and separation. The system processes employee data from input files, performs appropriate database operations, and maintains comprehensive audit and error logs.

### 1.2 Key Features

* Scheduled batch processing of employee data
* Support for multiple employee operations (Insert, Update, Delete)
* Transaction-based processing with data integrity
* Comprehensive error handling
* Complete audit trail
* Unique ID generation for logging

### 1.3 Business Context

The system facilitates Human Resources operations by: - Processing new employee records (recruitment) - Updating employee information, including promotions with salary adjustments - Managing employee separations - Maintaining historical records of all operations

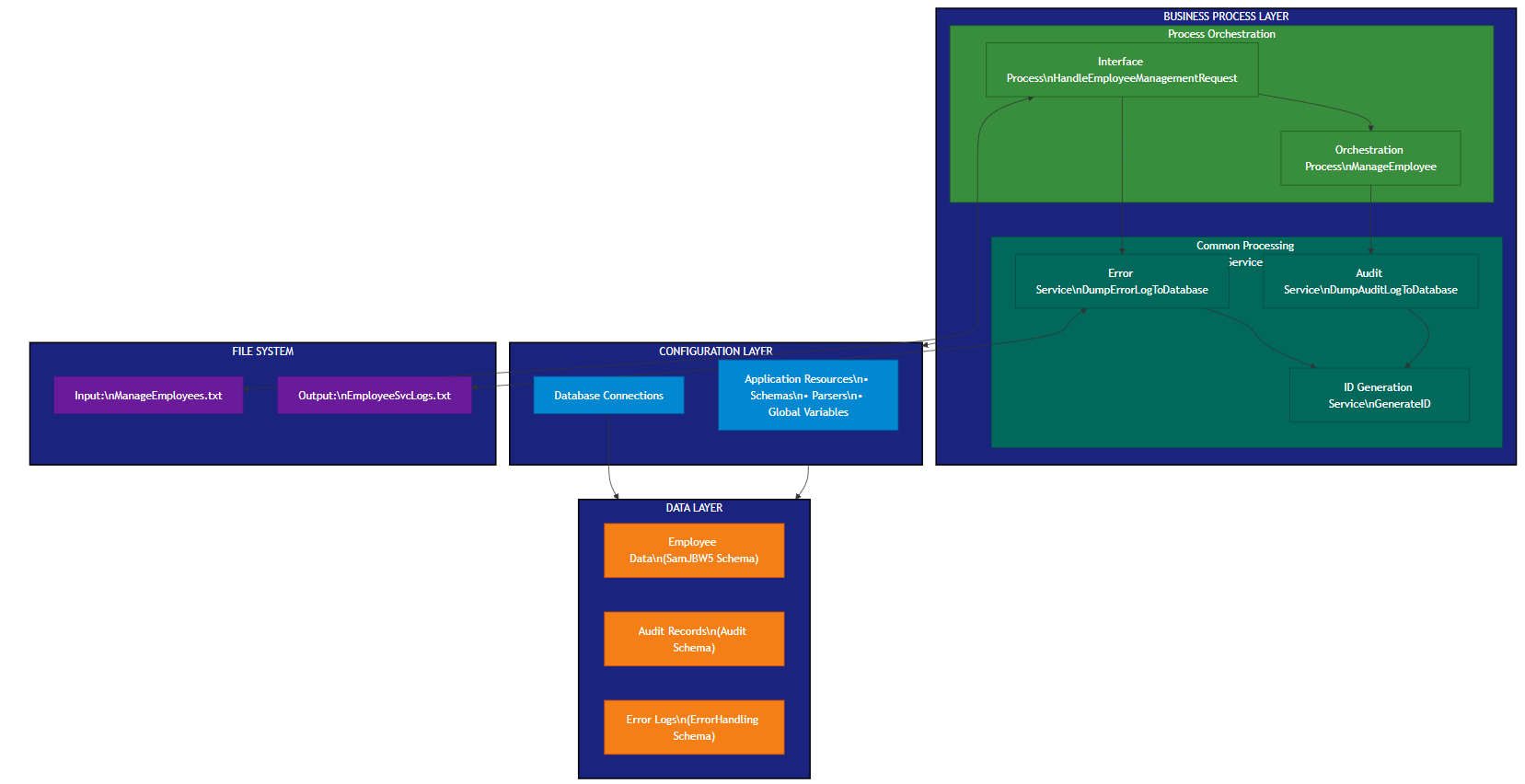
### 1.4 Technical Environment

* **Platform:** TIBCO BusinessWorks 5.x
* **Database:** MySQL 5.7 or higher
* **Environment:** Windows Server
* **Execution Mode:** Scheduled batch processing
* **File Format:** Comma-separated values (CSV)

## 2. Architecture

### 2.1 Architectural Overview

The Employee Management Service follows a layered architecture:



*Architecture Diagram*

#### 2.1.1 Business Process Layer

Contains the core application processes responsible for orchestrating employee management operations: - Interface Process (HandleEmployeeManagementRequest.process) - Orchestration Process (ManageEmployee.process) - Common Processes: - Audit Service (DumpAuditLogToDatabase.process) - Error Service (DumpErrorLogToDatabase.process) - ID Generation Service (GenerateID.process)

#### 2.1.2 Configuration Layer

Provides shared resources and configuration elements: - Database Connections - XML Schemas - Data Parsers - Global Variables

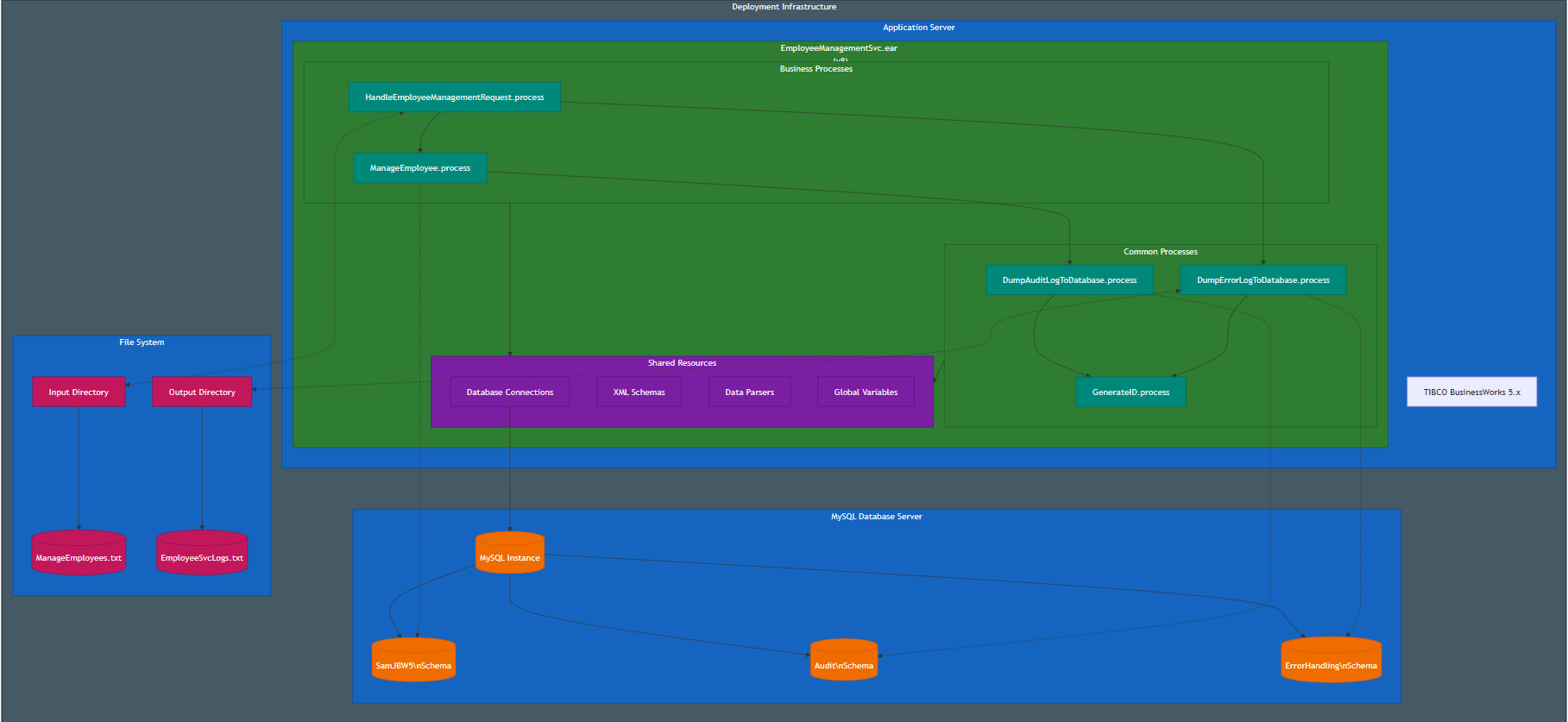
#### 2.1.3 Data Layer

Stores application data across three schemas: - Employee data (SamJBW5 schema) - Audit logs (Audit schema) - Error records (ErrorHandling schema)

#### 2.1.4 File System Layer

Manages input and output files: - Input file (ManageEmployees.txt) - Log file (EmployeeSvcLogs.txt)

### 2.2 Deployment Architecture



*Deployment Diagram*

The system is deployed as an Enterprise Archive (EAR) file on a TIBCO BusinessWorks 5.x runtime engine with the following components:

#### 2.2.1 Application Server

* TIBCO BusinessWorks 5.x Runtime
* EmployeeManagementSvc.ear (v8)
* Business and Common Processes
* Shared Resources

#### 2.2.2 Database Server

* MySQL Instance (localhost:3306)
* Three schemas:
  + SamJBW5: Employee records
  + Audit: Audit logs
  + ErrorHandling: Error records

#### 2.2.3 File System

* Input Directory: C:sam practice
* Output Directory: C:sam practice

### 2.3 Component Interactions

1. **Timer Trigger → Interface Process**
   * Daily scheduled execution at 10:00 PM
   * Initiates file processing
2. **Interface Process → Orchestration Process**
   * Passes file content for processing
   * Manages error scenarios
3. **Orchestration Process → Common Processes**
   * Calls Audit Service for successful operations
   * Both audit and error services use ID Generation Service
4. **Process → Database Interactions**
   * Orchestration Process operates on Employee Database
   * Audit Service writes to Audit Database
   * Error Service writes to Error Database
5. **File System Interactions**
   * Interface Process reads from input file
   * Error Process writes to log file

## 3. Component Specifications

### 3.1 Interface Process (HandleEmployeeManagementRequest.process)

**Purpose:** Entry point for the system, responsible for scheduling, file reading, and process orchestration.

**Key Activities:** - Timer (Scheduled Execution) - File Read (Input File) - Process Orchestration (Call ManageEmployee) - Error Handling (Multiple Levels) - Error Logging (File and Database)

**Process Variables:** - Error: String (Severe error type) - SystemError: String (System error type)

**Error Handlers:** - CatchAll: Handles all unhandled exceptions - Empty File Handler: Manages empty input files

**Dependencies:** - ManageEmployee.process - DumpErrorLogToDatabase.process - Global Variables (InputFile, LogEmployeeSvc)

### 3.2 Orchestration Process (ManageEmployee.process)

**Purpose:** Core business logic implementation for employee lifecycle management.

**Key Activities:** - Parse Data (CSV to XML) - Transaction Management (JDBC Transaction) - Employee Record Iteration - Database Operations: - RecruitEmployee (Insert) - PromoteEmployee (Update) - SeparateEmployee (Delete) - Employee Validation - Audit Logging

**Business Rules:** 1. Recruitment (Action=“I”): - Employee must not already exist - Default salary to 0 if not provided

1. Promotion (Action=“U”):
   * Employee must exist
   * Apply 2% salary increase
2. Separation (Action=“D”):
   * Employee must exist

**Dependencies:** - DumpAuditLogToDatabase.process - Database Connection (JDBCConnectionMSSQLDB) - Employee Schema

### 3.3 Audit Service (DumpAuditLogToDatabase.process)

**Purpose:** Records audit trail for employee operations.

**Key Activities:** - XML Rendering (Employee Details) - ID Generation (Call GenerateID) - Database Insert (Audit Records) - Error Handling (File Fallback)

**Audit Data Captured:** - Employee ID - Employee Name - Salary - Action Type (I/U/D) - Timestamp (Implicit) - Unique Audit ID

**Dependencies:** - GenerateID.process - Database Connection (JDBCConnectionAuditDB) - Global Variables (LogEmployeeSvc)

### 3.4 Error Service (DumpErrorLogToDatabase.process)

**Purpose:** Structured error logging for monitoring and troubleshooting.

**Key Activities:** - ID Generation (Call GenerateID) - Error Classification - Database Insert (Error Records) - File Logging (Fallback)

**Error Data Captured:** - Error Type - Stack Trace - Error Message - Class Information - Process Stack - Message Code - Timestamp (Implicit) - Unique Error ID

**Dependencies:** - GenerateID.process - Database Connection (JDBCConnectionErrorDB) - Global Variables (LogEmployeeSvc)

### 3.5 ID Generation Service (GenerateID.process)

**Purpose:** Creates unique identifiers for audit and error logs.

**Key Activities:** - UUID Generation (Java Activity) - String Conversion

**Implementation Details:** - Uses Java UUID.randomUUID() - Returns string representation of UUID

**Dependencies:** - None (self-contained)

## 4. Database Design

### 4.1 Database Overview

The application uses three MySQL database schemas:

1. **SamJBW5 Schema**: Main employee data
2. **Audit Schema**: Audit logging
3. **ErrorHandling Schema**: Error records

### 4.2 Database Schema Definitions

#### 4.2.1 SamJBW5 Schema

**Table: employee**

| Column | Type | Description | Constraints |
| --- | --- | --- | --- |
| empid | INTEGER | Employee ID | PRIMARY KEY |
| name | VARCHAR | Employee Name |  |
| salary | INTEGER | Employee Salary |  |

#### 4.2.2 Audit Schema

**Table: auditlog**

| Column | Type | Description | Constraints |
| --- | --- | --- | --- |
| auditlogid | VARCHAR | Unique Audit ID | PRIMARY KEY |
| inputmsg | VARCHAR | XML Representation of Operation |  |

#### 4.2.3 ErrorHandling Schema

**Table: dumperror**

| Column | Type | Description | Constraints |
| --- | --- | --- | --- |
| errorlogid | VARCHAR | Unique Error ID | PRIMARY KEY |
| errortype | VARCHAR | Error Classification |  |
| stacktrace | VARCHAR | Exception Stack Trace |  |
| msg | VARCHAR | Error Message |  |
| fullclass | VARCHAR | Error Source (Full Class) |  |
| class | VARCHAR | Error Source (Class Name) |  |
| ProcessStack | VARCHAR | Process Hierarchy |  |
| msgcode | VARCHAR | Error Code |  |

### 4.3 Database Connection Details

#### 4.3.1 Main Database

* **Connection Name:** JDBCConnectionMSSQLDB
* **Driver:** com.mysql.jdbc.Driver
* **URL:** jdbc:mysql://localhost:3306/SamJBW5
* **Username:** root
* **Password:** [Encrypted]
* **Max Connections:** 10
* **Login Timeout:** 0

#### 4.3.2 Audit Database

* **Connection Name:** JDBCConnectionAuditDB
* **Driver:** com.mysql.jdbc.Driver
* **URL:** jdbc:mysql://localhost:3306/Audit
* **Username:** root
* **Password:** [Encrypted]
* **Max Connections:** 10
* **Login Timeout:** 0

#### 4.3.3 Error Database

* **Connection Name:** JDBCConnectionErrorDB
* **Driver:** com.mysql.jdbc.Driver
* **URL:** jdbc:mysql://localhost:3306/ErrorHandling
* **Username:** root
* **Password:** [Encrypted]
* **Max Connections:** 10
* **Login Timeout:** 0

## 5. Process Flows

### 5.1 Main Process Flow

1. **Timer Trigger** (10:00 PM daily)
2. **Read Input File** from configured location
3. **Validate File Content**
   * If empty, log error and exit
   * If not empty, continue processing
4. **Parse Data** into employee records
5. **Begin Transaction**
6. **For Each Employee Record:**
   * Map employee details
   * Determine action type (I/U/D)
   * Validate business rules
   * Perform database operation
   * Log audit record
7. **Commit Transaction**
8. **Return Success** message

### 5.2 Error Handling Flow

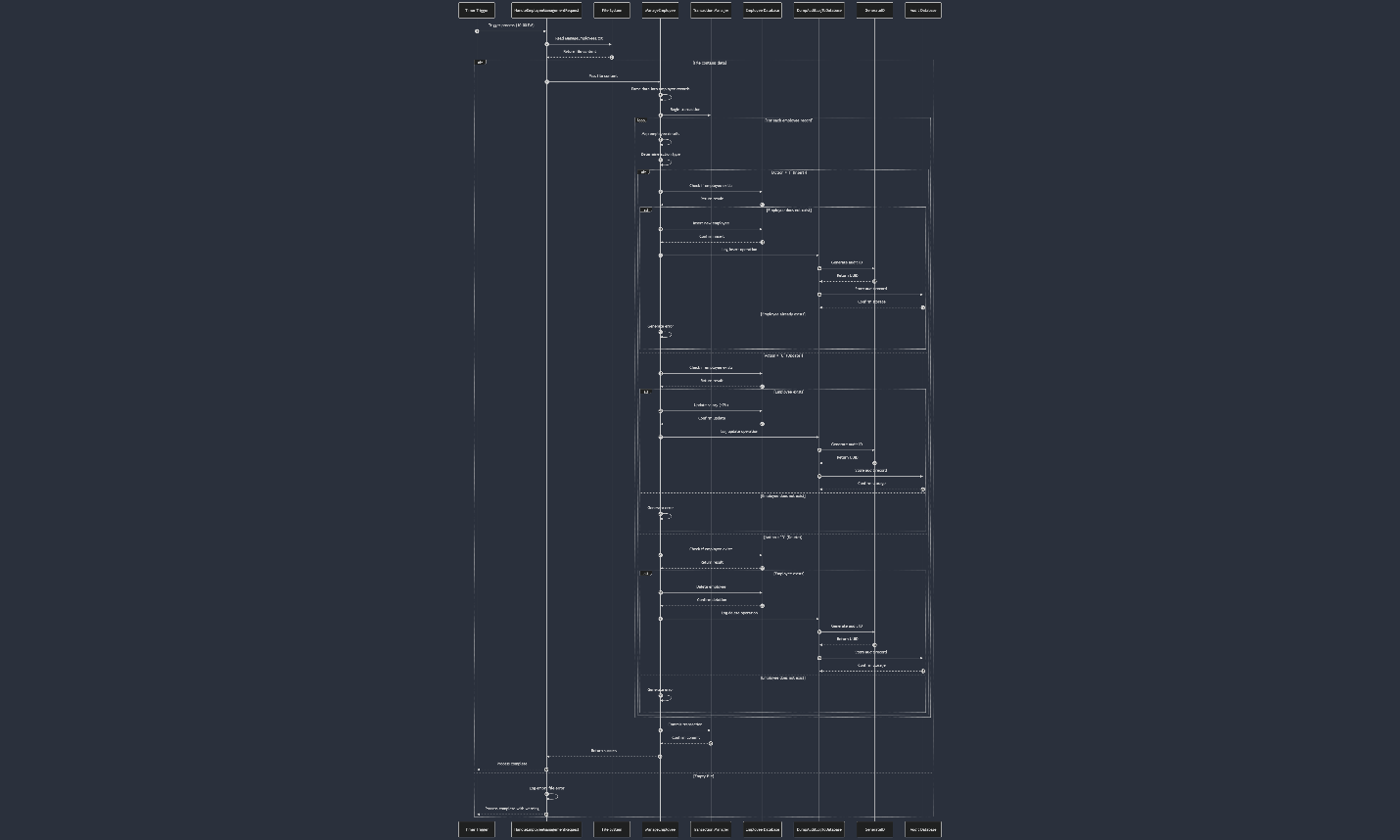
1. **Empty File Detection:**
   * Log to error file
   * Set error type to “SystemError”
   * Log to error database
   * Exit gracefully
2. **Unhandled Exceptions:**
   * Catch all exceptions
   * Log to error file
   * Set error type to “SevereError”
   * Log to error database
   * Log to system log
   * Exit gracefully
3. **Business Rule Violations:**
   * Generate specific error message
   * Exception propagates to Interface process
   * Handled by CatchAll handler

### 5.3 Audit Logging Flow

1. **Successful Operation**
   * Render employee details as XML
   * Generate unique audit ID
   * Insert audit record to database
   * Handle any exceptions during logging

### 5.4 Sequence Diagrams

#### 5.4.1 Normal Flow Sequence



*Normal Flow Sequence diagram*

#### 5.4.2 Error Handling Sequence

*Error Handling Sequence*

## 6. Error Handling

### 6.1 Error Classification

The system classifies errors into two main categories:

1. **System Errors**
   * Empty input files
   * Resource unavailability
   * Configuration issues
   * Environment-related issues
2. **Severe Errors**
   * Unhandled exceptions
   * Critical failures
   * Unexpected conditions
   * Runtime errors

### 6.2 Error Logging Mechanisms

#### 6.2.1 File Logging

* **Location:** C:sam practice.txt
* **Format:** Text with timestamp
* **Mode:** Append
* **Content:** Error message, source, timestamp

#### 6.2.2 Database Logging

* **Schema:** ErrorHandling
* **Table:** dumperror
* **Content:** Detailed error context including stack trace
* **ID Generation:** UUID-based

#### 6.2.3 System Logging

* **Mechanism:** TIBCO WriteToLog activity
* **Role:** User
* **Content:** Exception details

### 6.3 Error Recovery Strategies

1. **Transaction Rollback**
   * Automatic rollback on exception
   * Maintains database consistency
2. **Graceful Exit**
   * Proper error logging before termination
   * Clear error messages
3. **Fallback Logging**
   * File-based logging if database logging fails
   * Ensures error traceability

## 7. Configuration

### 7.1 Global Variables

**File: defaultVars.substvar**

| Variable Name | Value | Description |
| --- | --- | --- |
| InputFile | C:sam practice.txt | Input file path |
| LogEmployeeSvc | C:sam practice.txt | Log file path |
| Deployment | EmployeeManagementSvc | Deployment name |
| Domain | domain | Domain name |
| [Additional variables] | [Values] | System-level configurations |

### 7.2 Database Configurations

#### 7.2.1 Main Database (MSSQLDB/defaultVars.substvar)

| Variable Name | Value | Description |
| --- | --- | --- |
| DatabaseURL | com.mysql.jdbc.Driver | Driver class (misnamed) |
| JDCBDriver | jdbc:mysql://localhost:3306/SamJBW5 | Connection URL (misnamed) |
| UserName | root | Database username |
| Password | [Encrypted] | Encrypted password |

#### 7.2.2 Audit Database (AuditDB/defaultVars.substvar)

| Variable Name | Value | Description |
| --- | --- | --- |
| DatabaseURL | jdbc:mysql://localhost:3306/Audit | Connection URL |
| JDBCDriver | com.mysql.jdbc.Driver | Driver class |
| UserName | root | Database username |
| Password | [Encrypted] | Encrypted password |

#### 7.2.3 Error Database (ErrorDB/defaultVars.substvar)

| Variable Name | Value | Description |
| --- | --- | --- |
| DatabaseURL | jdbc:mysql://localhost:3306/ErrorHandling | Connection URL |
| JDBCDriver | com.mysql.jdbc.Driver | Driver class |
| UserName | root | Database username |
| Password | [Encrypted] | Encrypted password |

### 7.3 Schema Configurations

#### 7.3.1 Employee Schema (EmployeeDetails.xsd)

<xs:element name="EmployeeDetails">  
 <xs:complexType>  
 <xs:sequence>  
 <xs:element ref="emp\_id" minOccurs="0"/>  
 <xs:element ref="emp\_name" minOccurs="0"/>  
 <xs:element ref="salary" minOccurs="0"/>  
 <xs:element ref="Action"/>  
 </xs:sequence>  
 </xs:complexType>  
</xs:element>

#### 7.3.2 Error Schema (ErrorReport.xsd)

<xs:element name="ErrorReport">  
 <xs:complexType>  
 <xs:sequence>  
 <xs:element ref="ErrorType" minOccurs="0"/>  
 <xs:element ref="StackTrace" minOccurs="0"/>  
 <xs:element ref="Msg" minOccurs="0"/>  
 <xs:element ref="FullClass" minOccurs="0"/>  
 <xs:element ref="Class" minOccurs="0"/>  
 <xs:element ref="ProcessStack" minOccurs="0"/>  
 <xs:element ref="MsgCode" minOccurs="0"/>  
 </xs:sequence>  
 </xs:complexType>  
</xs:element>

#### 7.3.3 Audit Schema (AuditLogDetails.xsd)

<xs:element name="AuditLogDetails">  
 <xs:complexType>  
 <xs:sequence>  
 <xs:element ref="Input\_msg" minOccurs="0"/>  
 </xs:sequence>  
 </xs:complexType>  
</xs:element>

### 7.4 Parser Configurations

**File: EmployeeDetails.sharedparse** - **Format Type:** Delimiter separated - **Column Separator:** , (comma) - **Line Separator:** (newline) - **Fill Character:** Space - **Data Format:** References EmployeeDetails.xsd

## 8. Deployment

### 8.1 Deployment Package

**File: EmployeeManagementSvc.archive** - **Type:** Enterprise Archive (EAR) - **Version:** 8 - **Author:** prash - **Location:** C:.ear - **Main Process:** /Interface/HandleEmployeeManagementRequest.process

### 8.2 Deployment Structure

EmployeeManagementSvc.ear  
├── Shared Archive  
│ ├── Database Connections  
│ ├── XML Schemas  
│ ├── Parsers  
│ └── Global Variables  
└── Process Archive  
 ├── Interface Processes  
 ├── Service Processes  
 └── Common Processes

### 8.3 Deployment Prerequisites

1. **TIBCO BusinessWorks 5.x Runtime**
   * Minimum version: 5.11
   * Required patches: [List any specific patches]
2. **Database Requirements**
   * MySQL Server 5.7+
   * Three schemas: SamJBW5, Audit, ErrorHandling
   * Required tables created
   * User accounts with appropriate permissions
3. **File System Requirements**
   * Input directory with write permissions
   * Output directory with write permissions
   * Sufficient disk space for logs
4. **Java Requirements**
   * JRE 1.8 or higher
   * Required for UUID generation

### 8.4 Deployment Steps

1. **Database Setup**

* CREATE DATABASE SamJBW5;  
  CREATE DATABASE Audit;  
  CREATE DATABASE ErrorHandling;  
    
  USE SamJBW5;  
  CREATE TABLE employee (  
   empid INT PRIMARY KEY,  
   name VARCHAR(255),  
   salary INT  
  );  
    
  USE Audit;  
  CREATE TABLE auditlog (  
   auditlogid VARCHAR(40) PRIMARY KEY,  
   inputmsg TEXT  
  );  
    
  USE ErrorHandling;  
  CREATE TABLE dumperror (  
   errorlogid VARCHAR(40) PRIMARY KEY,  
   errortype VARCHAR(255),  
   stacktrace TEXT,  
   msg TEXT,  
   fullclass VARCHAR(255),  
   class VARCHAR(255),  
   ProcessStack TEXT,  
   msgcode VARCHAR(50)  
  );

1. **File System Setup**
   * Create directory: C:sam practice
   * Create directory: C:sam practice
   * Set appropriate permissions
2. **Deploy EAR File**
   * Use TIBCO Administrator
   * Navigate to Applications
   * Deploy EmployeeManagementSvc.ear
   * Verify deployment status
3. **Verify Configuration**
   * Check database connections
   * Verify file paths
   * Confirm application variables
4. **Start Application**
   * Enable scheduling
   * Monitor initial execution

## 9. Monitoring and Operations

### 9.1 Operational Procedures

#### 9.1.1 Daily Operations

1. **Input File Preparation**
   * Format: CSV with proper headers
   * Columns: emp\_id, emp\_name, salary, Action
   * Place in input directory by 9:30 PM
   * Ensure proper permissions
2. **Execution Monitoring**
   * Application runs at 10:00 PM daily
   * Check completion status in TIBCO Administrator
   * Verify log files for any issues
3. **Database Monitoring**
   * Check for new records in appropriate tables
   * Verify transaction completeness

#### 9.1.2 Maintenance Operations

1. **Log Rotation**
   * Archive logs older than 30 days
   * Maintain rolling 12 months of audit data
   * Clean up error database periodically
2. **Database Maintenance**
   * Run optimizations monthly
   * Check for table fragmentation
   * Verify indexes
3. **Application Maintenance**
   * Check for configuration drift
   * Verify connection pool settings
   * Monitor memory usage

### 9.2 Monitoring Points

#### 9.2.1 Application Monitoring

* TIBCO Process status in Administrator
* Process execution time
* Success/failure status
* Resource consumption

#### 9.2.2 Database Monitoring

* Connection pool utilization
* Query performance
* Table growth
* Error frequency

#### 9.2.3 File System Monitoring

* Input file arrival
* Output file growth
* Disk space utilization
* File permissions

#### 9.2.4 Business Metrics

* Number of employees processed
* Operation distribution (I/U/D)
* Error rates by type
* Processing completion time

### 9.3 Alerts and Notifications

#### 9.3.1 Critical Alerts

* Process failure
* Database connection failure
* Empty input file
* Severe errors

#### 9.3.2 Warning Alerts

* Processing time exceeding threshold
* High error rate
* File system space issues
* Database performance degradation

## 10. Troubleshooting Guide

### 10.1 Common Issues and Resolutions

#### 10.1.1 Empty File Error

**Symptoms:** - Process completes quickly - “Empty file” entry in error log - No database updates

**Resolution:** 1. Verify input file exists and has content 2. Check file permissions 3. Ensure file format is correct 4. Review global variable for correct path

#### 10.1.2 Database Connection Issues

**Symptoms:** - Database connection errors in logs - Process terminates abnormally - Severe errors recorded

**Resolution:** 1. Verify MySQL service is running 2. Check connection parameters 3. Test database connectivity independently 4. Ensure user credentials are correct 5. Verify network connectivity

#### 10.1.3 Transaction Failures

**Symptoms:** - Partial processing - Rollback messages in logs - Integrity constraint violations

**Resolution:** 1. Check input data integrity 2. Verify business rules compliance 3. Ensure database schema is correct 4. Check for conflicting operations

#### 10.1.4 Process Timeout

**Symptoms:** - Process runs longer than expected - Timeout messages in logs - Incomplete processing

**Resolution:** 1. Check database performance 2. Verify input file size is reasonable 3. Review transaction boundaries 4. Check for database locks or contention

### 10.2 Error Log Analysis

#### 10.2.1 System Error Analysis

1. Check EmployeeSvcLogs.txt for initial error
2. Query ErrorHandling.dumperror for detailed context
3. Look for “SystemError” type entries
4. Review file system and resource availability

#### 10.2.2 Severe Error Analysis

1. Check EmployeeSvcLogs.txt for exception details
2. Query ErrorHandling.dumperror for stack trace
3. Look for “SevereError” type entries
4. Analyze process stack to identify failure point

#### 10.2.3 Business Rule Violations

1. Look for specific error messages
2. Check input data against business rules
3. Verify employee existence for updates/deletes
4. Confirm data format compliance

### 10.3 Recovery Procedures

#### 10.3.1 File Recovery

1. Restore input file from backup if corrupted
2. Check for partial processing
3. Consider rerunning with corrected file
4. Verify file format and encoding

#### 10.3.2 Database Recovery

1. Check transaction logs for partial commits
2. Verify database consistency
3. Restore from backup if necessary
4. Run validation queries to confirm integrity

#### 10.3.3 Process Recovery

1. Check process state in TIBCO Administrator
2. Restart application if necessary
3. Manually trigger process after fixing issues
4. Monitor closely for successful completion

## Appendices

### Appendix A: Input File Format

**File Name:** ManageEmployees.txt **Format:** Comma-separated values (CSV) **Encoding:** ASCII

**Structure:**

emp\_id,emp\_name,salary,Action  
1001,John Doe,75000,I  
1002,Jane Smith,82000,U  
1003,Bob Johnson,68000,D

**Field Descriptions:** - **emp\_id**: Integer employee identifier - **emp\_name**: String employee name - **salary**: Integer salary amount - **Action**: Single character action code - I: Insert (new employee) - U: Update (promotion) - D: Delete (separation)

**Rules:** 1. Headers optional 2. emp\_id required for updates and deletes 3. emp\_name required for inserts 4. salary optional (defaults to 0) 5. Action required (I, U, or D)

### Appendix B: Process Performance Metrics

| Process | Average Duration | Peak Duration | Resource Usage |
| --- | --- | --- | --- |
| HandleEmployeeManagementRequest | 2-5 minutes | 15 minutes | Low |
| ManageEmployee | 1-4 minutes | 12 minutes | Medium |
| DumpAuditLogToDatabase | <1 second | 2 seconds | Low |
| DumpErrorLogToDatabase | <1 second | 2 seconds | Low |
| GenerateID | <0.1 second | 0.2 seconds | Very Low |

### Appendix C: Version History

| Version | Date | Changes | Author |
| --- | --- | --- | --- |
| 1.0 | [Initial Date] | Initial implementation | prash |
| … | … | … | … |
| 8.0 | [Current Date] | Current production version | prash |

*End of Documentation*