

---

## **EDI Batch Process**

---

## **CONTENTS**

<b>1</b>	<b>Purpose .....</b>	<b>3</b>
1.1	Use Case – EDI service .....	3
1.2	Use Case – EDI Daily Reporting .....	3
1.3	Use Case – EDI Service Monitoring Process .....	3
<b>2</b>	<b>EDI Process Design – High Level .....</b>	<b>4</b>
2.1	EDI Batch Service .....	4
2.1.1	Message FTP Service .....	4
2.1.2	Message Processing Service .....	4
2.1.3	Response Service .....	4
2.2	EDI Batch Monitoring Service .....	4
2.3	EDI Batch scheduling service .....	4
2.4	EDI Batch reporting Service .....	4
<b>3</b>	<b>EDI Process Design –Detailed .....</b>	<b>5</b>
3.1	Message FTP Service .....	5
3.2	Message Processing Service .....	6
3.3	Response Service .....	7
3.4	Batch Process Monitoring Service .....	8
3.5	Scheduling Service .....	9
3.6	EDI reporting process .....	9
3.7	Managing Failure Conditions .....	9
3.8	Batch Process Maintenance .....	9
3.9	Purge Requirements .....	9
<b>4</b>	<b>Development/Production Environment .....</b>	<b>10</b>
4.1	Development Environment .....	10
4.2	Runtime Environment .....	10
<b>5</b>	<b>Approach and Environment .....</b>	<b>10</b>
5.1	Exception handling .....	10
5.2	Logging .....	11
5.3	Architectural Goals and Constraints .....	12
<b>Appendix A – List of Log MESSAGES .....</b>		<b>13</b>
<b>Appendix B –Configuration Parameters .....</b>		<b>14</b>
<b>Appendix C– Batch Database Design .....</b>		<b>15</b>

## **1 Purpose**

The purpose of this document is to discuss the design of the MITCO EDI Service. This service is responsible for processing incoming and outgoing EID/CSV/FLAT File messages.

### **1.1 Use Case – EDI service**

The MITCO EDI Service is a batch processes which runs indefinitely. It receives the message files from different clients via FTP and processes them into database. It also sends out the response if required.

### **1.2 Use Case – EDI Daily Reporting**

This has to be discussed. Idea is to generate a report of EDI activity in the systems on a daily basis and send it out to required people. Also this can be used for the acknowledgement of the received messages.

### **1.3 Use Case – EDI Service Monitoring Process**

This process will on a regular interval check that the EDI service is running and functioning properly. In case of errors notifications will be sent to appropriate person.

## **2 EDI Process Design – High Level**

The following sections discuss the high-level architecture and software architecture for the MITCO EDI Process. EDI process mainly consists of three main services -

### **2.1 EDI Batch Service**

EDI Batch service is a java based batch process which utilizes **Spring Batch Framework**. EDI Service is the heart of EDI process and completes three major functions -

#### **2.1.1 Message FTP Service**

FTP the message files to the required locations from FTP servers – FTP process uses the java FTP API to move the file from FTP locations to local machine where batch process is running. FTP process also backup the files and

#### **2.1.2 Message Processing Service**

Process the message files and update the Database

#### **2.1.3 Response Service**

Send out the EDI responses

### **2.2 EDI Batch Monitoring Service**

This process will be a separate process and will be scheduled to run through windows scheduling service. This process on a regular interval will check the health of EDI main batch process.

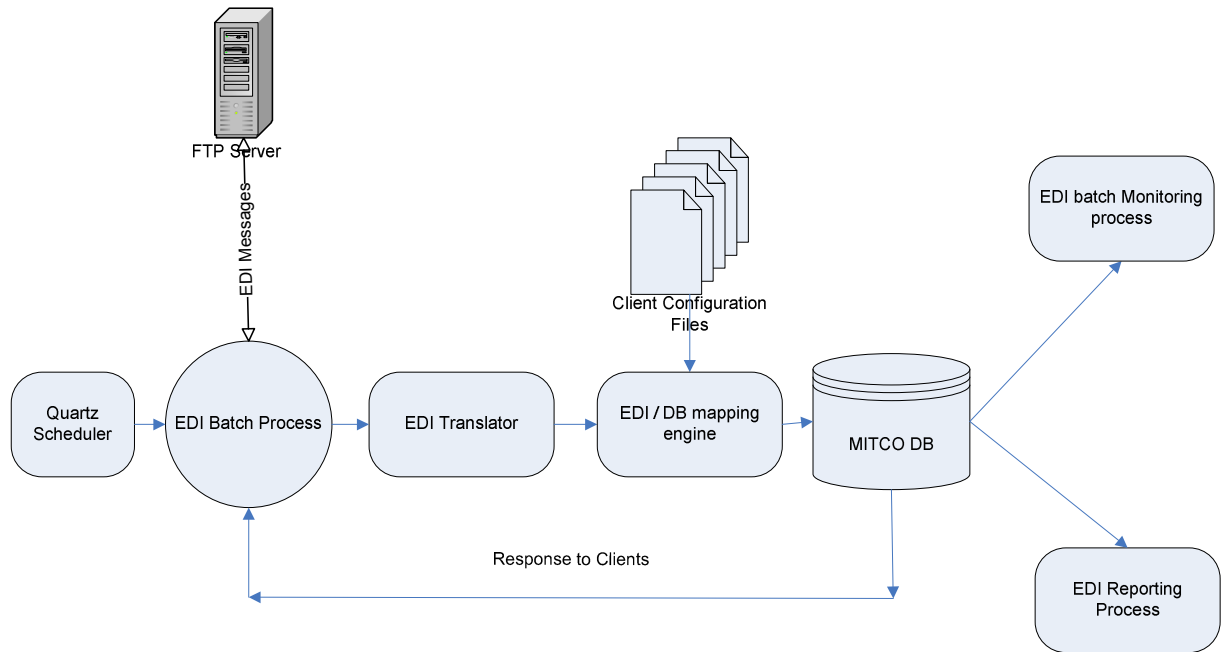
### **2.3 EDI Batch scheduling service**

This service will be responsible for executing the Batch process at regular intervals.

### **2.4 EDI Batch reporting Service**

This process generates a daily EDI activity report. Consolidating the number of messages received from different clients etc.

Following is a high-level diagram of the components discussed in this design-



### 3 EDI Process Design –Detailed

#### 3.1 Message FTP Service

FTP service will be responsible for getting the files from FTP servers of clients and MITCO.

##### Components:

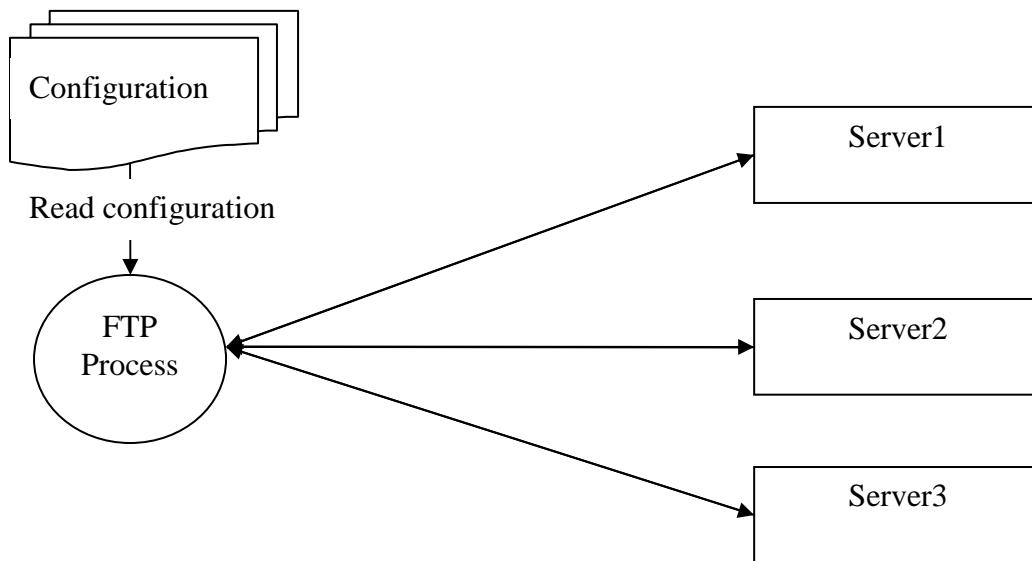
- 1) **Configuration file** – This file will have configurations for different servers from which file will be received. A sample of the configuration file is as below –

[ftp.mitcoltd.com|userid|password|client|RemoteDirectory|LocalDirectory|file\\_name\\_patten](ftp.mitcoltd.com|userid|password|client|RemoteDirectory|LocalDirectory|file_name_patten)

For adding a new FTP location (Addition of a client) the responsible person will have to make an entry to the configuration file. FTP program will read the new configuration and automatically start downloading the files from the remote server.

- 2) **Java based FTP program** – FTP program will read the configuration file and use java FTP API to move the files from FTP locations to the local machines. FTP program will –

- 1) Move the files to the local machine based on the configuration file
- 2) Put the files in backup directories
- 3) Make appropriate entries in the database
- 4) In case of any errors it will notify the responsible person.
- 5) If one or more FTP servers are unavailable then appropriate notifications will be sent, but the process will execute for the servers which are available.



### 3.2 Message Processing Service

Message Processing Service is responsible for processing the EDI/CSV messages and store them in database.

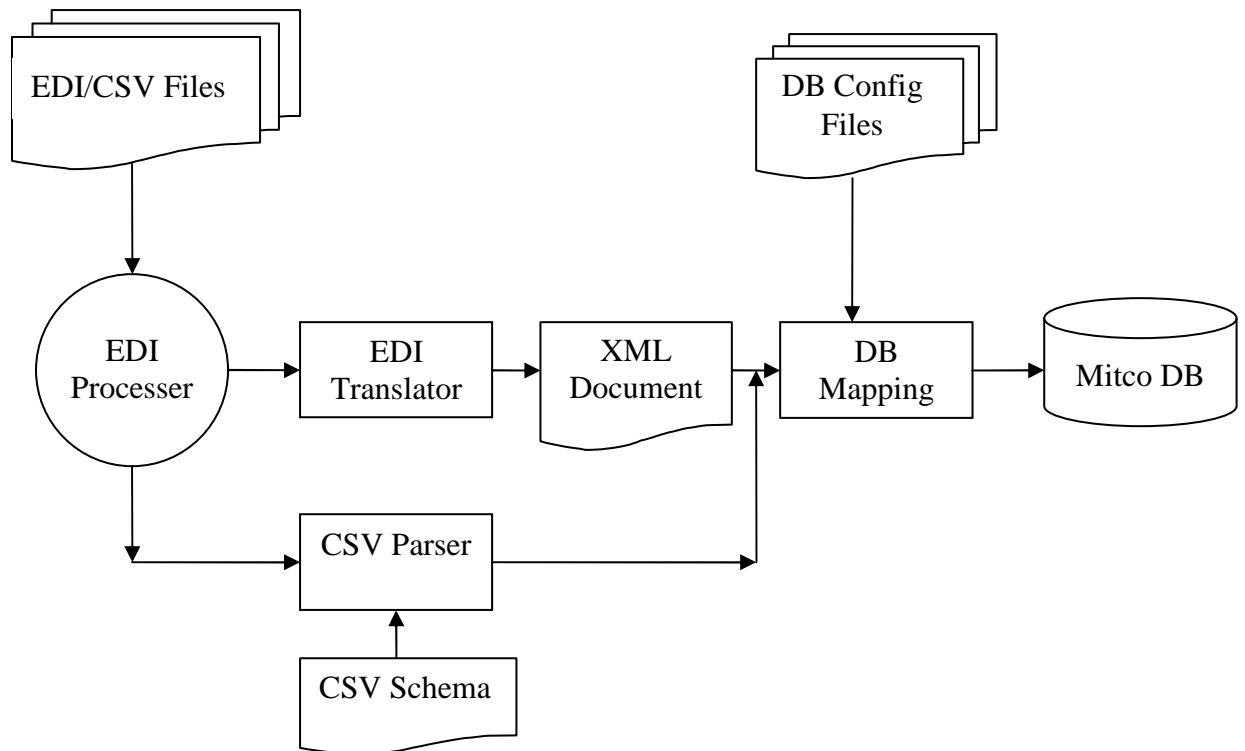
#### Components:

- 1) **DB Configuration files** – These files will be specific to clients. These files will map the data from XML files to DB. Similar mappings will be used for CSV data to database. For a new message addition the corresponding DB configuration file will be created.
- 2) **CSV schema files** – These files will be used to parse the CSV files. The configuration of all the CVS messages will be in a single file. If a new CSV message is added the corresponding schema will be added in the schema file. Processing program will automatically pick the schema based on the message type. Schema files will look as below

–

```

    MessageType1|Field1|Field2|Field3|Field4|Field5|Field6|Field7|Field8|Field9
    MessageType2|Field1|Field2|Field3|Field4|Field5|Field6|Field7|Field8|Field9
    MessageType3|Field1|Field2|Field3|Field4|Field5|Field6|Field7|Field8|Field9
  
```



- 3) **EDI Translator** – EDI translator is a third party tool and will be responsible for translating EDI messages in XML format. XML documents created will be mapped to database.
- 4) **CSV Parser** – CSV parser will parse the incoming CSV files and then the Data will be mapped to Database fields with the help of DB configuration files.
- 5) **EDI processor** – This is the main program which will execute the processing of EDI/CSV files. Program will use the following logic.
  - a. Identify from database the files which have to be processed
  - b. Read files one by one from local drive. If the file is an EDI message go to step c. If the files are CSV then go to step d
  - c. Pass the file to EDI translator and generate the XML file. Based on the type of message and client, identify the database mappings and insert the data into database.
  - d. Pass the CSV to the CSV parser. Based on the type of message and client, identify the database mappings and insert the data into database.

### 3.3 Response Service

This service will generate the EDI responses to be sent to different clients –

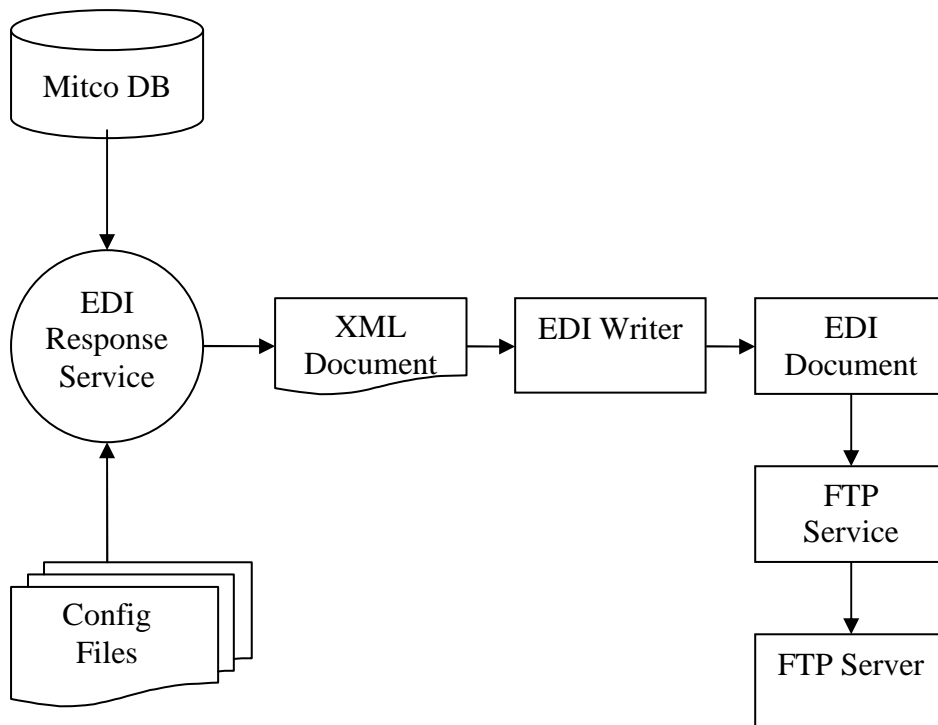
#### Components:

- 1) **Configuration files** – These configuration files will contain the information about the responses to be sent out to different clients. If a new response is to be sent to a client then

the configuration files need to be updates. Similarly if the response is to be stopped then the configuration file needs to be updated. A sample configuration file will look as below -

CustCode|TypeOfResponse|Y|ftp.mitcoltd.com|username|password|remoteDir

- 2) **EDI writer** – This is a third party tool and is responsible for translating the XML files into EDI messages. EDI messages will be transferred to different clients using the FTP service.
- 3) **Java based FTP program** – FTP program will move the files from local machine to client FTP locations.
- 4) **EDI Response Service** – EDI response service will be responsible for following –
  - a. Extracting information from Database and translating it into an XML file. If a new message is added to the configuration file then the additional code will be required to write the XML file. (This will not be a difficult process; just the logic will have to be manipulated a bit).
  - b. Call the EDIWriter API to translate the message into EDI format.
  - c. FTP the file to the desired location.
  - d. Make appropriate entries into the database.
  - e. In case of failure send out the appropriate notifications.



### 3.4 Batch Process Monitoring Service

This service can be used to monitor the activity of batch process at any given point of time. This service will check the health of batch process at regular interval of time. If an error is detected then it will send out alerts. This program will be scheduled through windows

scheduling service, this will make sure that this program executed at regular interval of time no matter what.

The actual functionality will be discussed later while development is going on.

### **3.5 Scheduling Service**

This service will be used to schedule the execution of batch process. This service will be based on freely available framework called Quartz.

### **3.6 EDI reporting process**

This service can be used to monitor the activity of batch process on a daily basis. It can consolidate several reports to view.

### **3.7 Managing Failure Conditions**

- 1) **Unavailability of Database** – If the Database is unavailable then the batch process will suspend any activity and send out alerts. As soon as the DB is available the batch process will recover from the point it failed. If a file was being processed and DB went down then the file will be processed again.
- 2) **General Failure in the process**- If there were any general failure in the batch process, it will recover automatically. If the error can't be recovered automatically then appropriate alerts will be sent.
- 3) **Files not processed** – if a file was missed or was not processed properly then that file can be added to a configuration file and the batch process will process it in the very next cycle.
- 4) **Failure in sending the response** – It can happen due to several reasons. E.g. network not available, FTP servers not available etc. In all these cases batch process will queue the responses. Appropriate alerts will be sent. As soon as the condition is corrected Batch process will automatically start sending out the messages to specific FTP sites.
- 5) **Logs** – Logs can be used for troubleshooting

### **3.8 Batch Process Maintenance**

If for some reason the batch process has to be stopped, then this should be planned ahead of time. StopConfigFile should be updated for stopping the batch process at a specific time of day. Batch Process will stop the basic functionality at this point of time. Process can be killed or stopbatch.bat can be executed to stop the process at this point of time.

For starting batch process, startbatch.bat batch file should be executed. Batch process will send out a notification after it has been started.

### **3.9 Purge Requirements**

The Purge Utility will be responsible for purging log files written out by the batch process after 30 days.

## 4 Development/Production Environment

### 4.1 Development Environment

#### Software:

Front End PC Requirements:

- Typical installation of Windows XP SP2 and above.
- Development tools – Eclipse 3.3

Back End Requirements:

- JDK 6.0
- SQL server 2000
- Spring Batch Framework

**Defect Control Tools:**

- Excel files

### 4.2 Runtime Environment

#### Software:

Back End Requirements:

- JDK 6.0
- SQL server 2000
- Spring Batch Framework

## 5 Approach and Environment

### 5.1 Exception handling

**Unless otherwise specified the exception handling will be done as per the following:**

All classes called from the session beans should either throw specific application exceptions (e.g. EDISystemException) on their own or should propagate the exceptions generated by the methods they are calling by simply mentioning them in their throws clause.

Avoid catching and then re-throwing the same exception in any classes. This forces the overhead of multiple exception stacks being created; impacting the runtime performance.

Exception will be defined in two major categories:

- 1) EDISystemException: Will capture system level exceptions. System level exceptions will capture all system failure related issues. E.g. service not available.
- 2) EDIBusinessException: Will capture business level exceptions. Business level exception will capture all business related issues.

## 5.2 Logging

The log file will be created to log the detailed messages generated by EDI process. There will be one log file generated for the entire application with the datestamp. The log file opens in read + write mode. If the file already exists it will open in write mode and if it doesn't exist then a new file will be created with write permissions. The log file name is taken from database -

The naming convention used for Log file generated is following: -

**<AppName>Log.<mmddy>**

Where,

<AppName> is the name of application

<Log> indicating that the file is general log file

<mmddy> is the date stamping done on the log file. 'mmddy' denotes 2-digit month figure , 2-digit day and 2 -digit year figure.

The following will be the information passed by the processes while requesting for logging message.

- Message Severity - The severity level passed by the process.
- Origin Class Name - Name of the class where the log message originated
- Origin Method Name - The name of the method where the log message originated
- Message Text - Message text for the message logged. The message text describes the event logged by the user.

The severity level passed above by the calling process will be checked against the LOG\_LEVEL value stored in the Directory server and only those messages will be logged for which the severity is greater than equal to Log Level setting.

The following are the things logged into the log file:

- 1) Logging Date & time - Logging date time value obtained at the time of logging.
- 2) Severity - The severity of the Message/Error logged.
- 3) Origin Class - The class where the Log Message originated.
- 4) Message details- The details of the message/error logged.

The severity of logging levels for logging is as follows:

Severity Code	Severity Classification	Description
LOG_CRITICAL_ERR	9	Database errors
LOG_ERR	6	Error & No Conformance Business Rule
LOG_PROC_STATUS	5	Start and End process
LOG_FIELD_VAL	4	Not a valid entry in the field.
LOG_NORMAL	3	Normal Flow
LOG_LOGIC	2	Logic Flow
LOG_DEBUG	1	For all other messages

### **5.3 Architectural Goals and Constraints**

Batch architecture is designed for the following goals:

#### **Scalability**

Scalability implies the ability for the architecture to grow and accommodate increasing numbers of users, applications, and systems. The architecture must also accommodate changing business processes, migrating data, and new uses of existing data.

#### **Flexibility**

Flexibility implies the ability for the architecture to remain vendor and technology independent. Technology will always influence the structure of the architecture but in no means should constrain the architecture.

#### **Extensibility**

Extensibility implies the ability for the architecture to allow new functionality to be added. It should grow and accommodate new business requirements as they are determined and incorporated into the architecture.

#### **Reusability**

The architecture fosters reuse wherever possible. Equally important is the underlying effect that implementation of the architecture encourages the development of reusable components.

## **Appendix A – List of Log MESSAGES**

All the log messages will be accompanied by a code. E.g.

Log messages will be provided when they are finalized in development process.

## Appendix B – Configuration Parameters

The following section provides information on the set-up of the **Batch configuration parameters**.

### Application Information

<b>Application Name</b>	
<b>Application Version</b>	
<b>Application Host</b>	
<b>Application Instance</b>	

### Application Configuration

<b>Configuration Item</b>	<b>Configuration Value (Examples)</b>	<b>Description</b>	<b>Comment (if any)</b>

Note: More parameters may be added to the list while development.

## Appendix C– Batch Database Design

--Autogenerated: do not edit this file

```
DROP TABLE BATCH_EXECUTION_CONTEXT ;
DROP TABLE BATCH_STEP_EXECUTION ;
DROP TABLE BATCH_JOB_EXECUTION ;
DROP TABLE BATCH_JOB_PARAMS ;
DROP TABLE BATCH_JOB_INSTANCE ;
DROP TABLE BATCH_STEP_EXECUTION_SEQ ;
DROP TABLE BATCH_JOB_EXECUTION_SEQ ;
DROP TABLE BATCH_JOB_SEQ ;
```

```
CREATE TABLE BATCH_JOB_INSTANCE (
    JOB_INSTANCE_ID BIGINT NOT NULL PRIMARY KEY ,
    VERSION BIGINT ,
    JOB_NAME VARCHAR(100) NOT NULL,
    JOB_KEY VARCHAR(2500)
);
```

```
CREATE TABLE BATCH_JOB_EXECUTION (
    JOB_EXECUTION_ID BIGINT NOT NULL PRIMARY KEY ,
    VERSION BIGINT ,
    JOB_INSTANCE_ID BIGINT NOT NULL,
    CREATE_TIME DATETIME NOT NULL,
    START_TIME DATETIME DEFAULT NULL ,
    END_TIME DATETIME DEFAULT NULL ,
    STATUS VARCHAR(10) ,
    CONTINUABLE CHAR(1) ,
    EXIT_CODE VARCHAR(20) ,
    EXIT_MESSAGE VARCHAR(2500) ,
    constraint JOB_INST_EXEC_FK foreign key (JOB_INSTANCE_ID)
    references BATCH_JOB_INSTANCE(JOB_INSTANCE_ID)
);
```

```
CREATE TABLE BATCH_JOB_PARAMS (
    JOB_INSTANCE_ID BIGINT NOT NULL ,
    TYPE_CD VARCHAR(6) NOT NULL ,
    KEY_NAME VARCHAR(100) NOT NULL ,
    STRING_VAL VARCHAR(250) ,
    DATE_VAL DATETIME DEFAULT NULL ,
    LONG_VAL BIGINT ,
    DOUBLE_VAL DOUBLE PRECISION ,
    constraint JOB_INST_PARAMS_FK foreign key (JOB_INSTANCE_ID)
    references BATCH_JOB_INSTANCE(JOB_INSTANCE_ID)
);
```

```
CREATE TABLE BATCH_STEP_EXECUTION (
    STEP_EXECUTION_ID BIGINT NOT NULL PRIMARY KEY ,
    VERSION BIGINT NOT NULL,
    STEP_NAME VARCHAR(100) NOT NULL,
    JOB_EXECUTION_ID BIGINT NOT NULL,
```

```
START_TIME DATETIME NOT NULL ,
END_TIME DATETIME DEFAULT NULL ,
STATUS VARCHAR(10) ,
COMMIT_COUNT BIGINT ,
ITEM_COUNT BIGINT ,
READ_SKIP_COUNT BIGINT ,
WRITE_SKIP_COUNT BIGINT ,
ROLLBACK_COUNT BIGINT ,
CONTINUABLE CHAR(1) ,
EXIT_CODE VARCHAR(20) ,
EXIT_MESSAGE VARCHAR(2500) ,
constraint JOB_EXEC_STEP_FK foreign key (JOB_EXECUTION_ID)
references BATCH_JOB_EXECUTION(JOB_EXECUTION_ID)
);

CREATE TABLE BATCH_EXECUTION_CONTEXT (
EXECUTION_ID BIGINT NOT NULL,
DISCRIMINATOR VARCHAR(1) NOT NULL,
TYPE_CD VARCHAR(6) NOT NULL,
KEY_NAME VARCHAR(1000) NOT NULL,
STRING_VAL VARCHAR(1000) ,
DATE_VAL DATETIME DEFAULT NULL ,
LONG_VAL BIGINT ,
DOUBLE_VAL DOUBLE PRECISION ,
OBJECT_VAL IMAGE
);

CREATE TABLE BATCH_STEP_EXECUTION_SEQ (ID BIGINT IDENTITY);
CREATE TABLE BATCH_JOB_EXECUTION_SEQ (ID BIGINT IDENTITY);
CREATE TABLE BATCH_JOB_SEQ (ID BIGINT IDENTITY);
```