OLE Batch Process Technical Documentation

- Purpose
- Dependencies (db tables)
- Logical Data Model (Class Structure)
- Physical Data Model (Database Schema)
- Service Interface Design (Java)
- Flowchart
  - Order Record Import
  - Invoice Import
  - Batch Delete
  - Batch Export
  - Patron / Location Import
  - Bib Import
  - Claim Report
  - Serial Record Import
- Pseudo Code
  - Order Record Import
  - Invoice Import
  - Batch Delete
  - Batch Export
    - Logic behind File Name
  - Patron Import
  - Bib Import
    - Matching Process
  - Location Import
  - Claim Report
  - Serial Record Import
- Sample XML
  - Patron
  - Location
  - Order
    - EDI Sample
    - MARC Sample
- Service Interface Design (SOAP/REST)
- User Interface Design
- Data Importing
- Data Exporting (if applicable)
- Workflow
- System Parameters
- Roles and Permissions

Last Published December 8, 2015

Purpose

Batch Process allows to make use of Batch Process Profiles and help to import and/or to export large chunks of data viz. Bibliographic information, Order records, Invoices, Patron records, Locations, Serial records and claim report. In addition, it can be used to delete Bib records.

Dependencies (db tables)

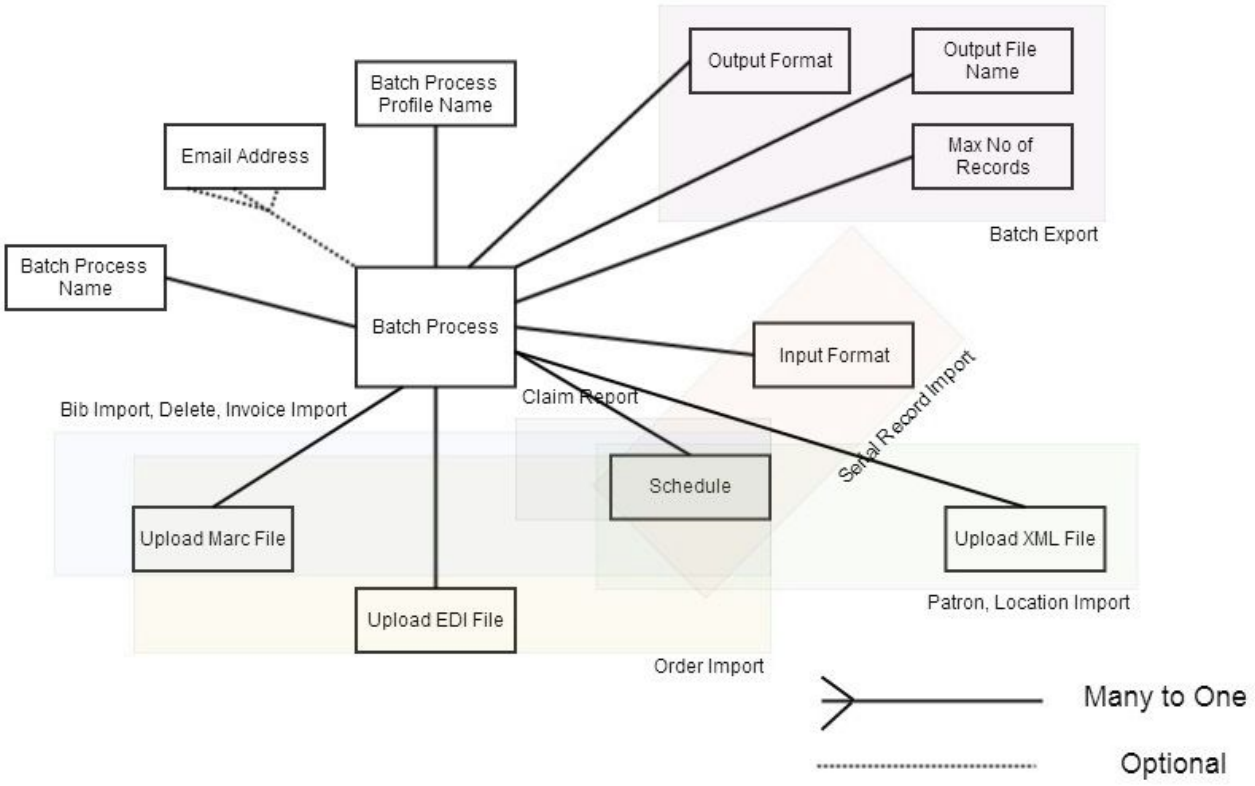
Tables used in Batch Process creation

ole_bat_prcs_prf_t	Batch Process Profile

Tables where Batch Process is used/referenced

ole_bat_prcs_job_t	Batch Process Job
ole_bat_prcs_schdule_t	Batch Process Schedule

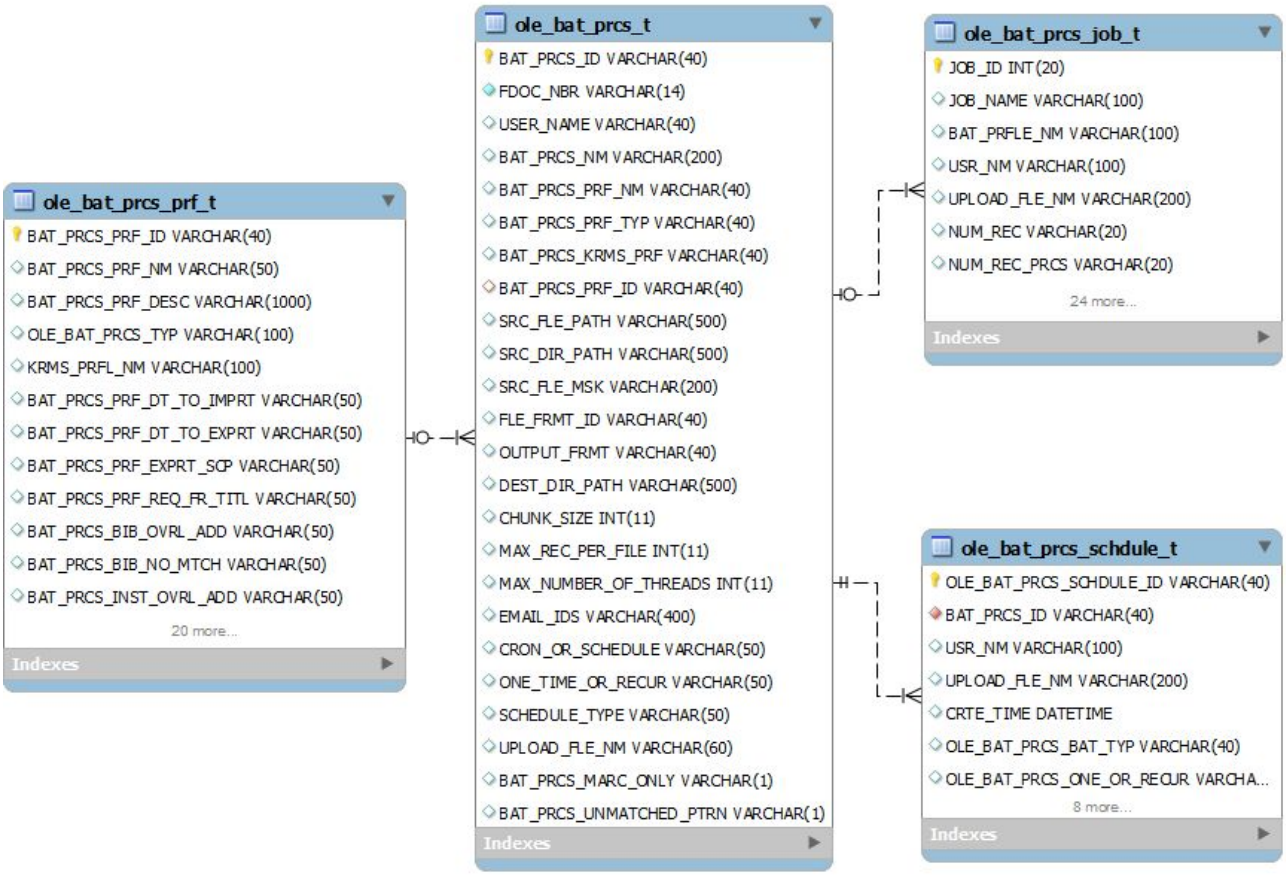
Logical Data Model (Class Structure)



Depending on the Batch Process Type, various different Batch Processes need various data. By default, a Batch Process will have a name and profile attached to it. Email address to whom report should be sent is optional and can take multiple email ids. In case of Patron and Location Import, the file to be uploaded is XML and the process can be scheduled. In case of Order Import, a MARC file, an EDI file and schedule has to be set while in case of Bib import, delete and Invoice Import, a MARC file and schedule information is enough. When doing a Serial record import input format can

be specified along with schedule and based on the format file upload can be done. For Batch export, the output file name, format and max number of records in each file has to be specified.

Physical Data Model (Database Schema)

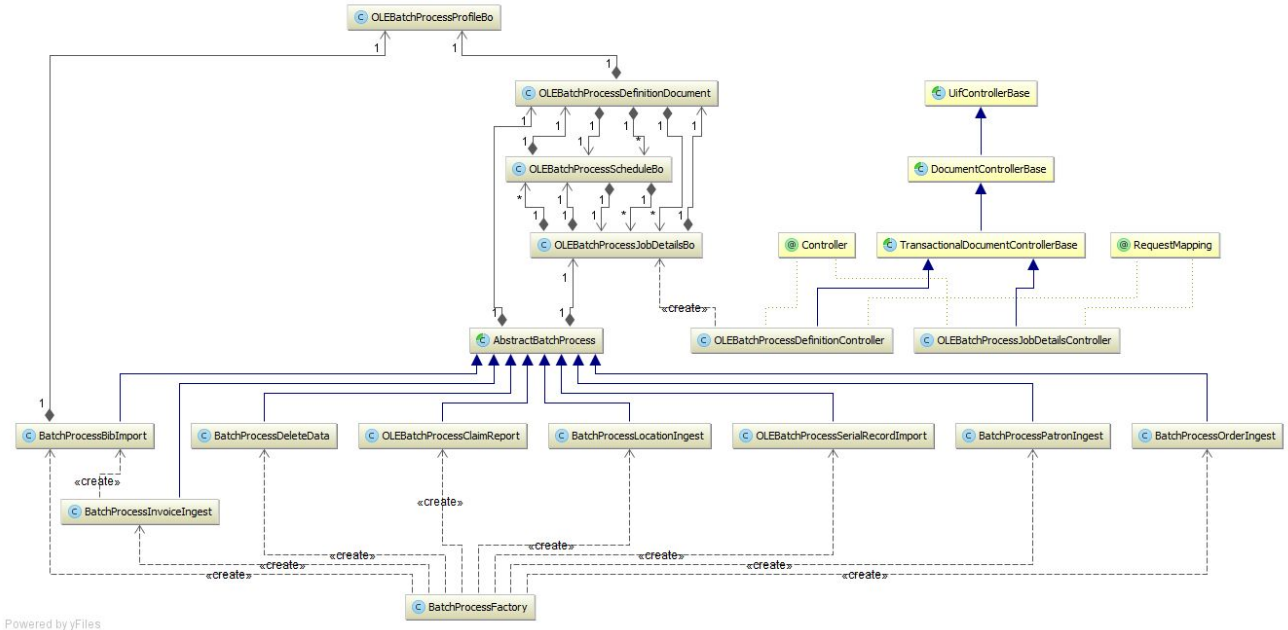


The Batch Process (*ole_bat_prcs_t*) will have one Batch Process Profile (*ole_bat_prcs_prf_t*) linked to it. However, multiple Batch Processes can use the same Batch Process Profile. When the Batch Process is scheduled then it goes into the *ole_bat_prcs_schdule_t* table while the job that is finished running is added to the *ole_bat_prcs_job_t* table. Both these tables reference the Batch Process Id (*BAT_PRCS_ID*) from the *ole_bat_prcs_t* table.

Service Interface Design (Java)

Transaction Documents are used to create Batch Processes. The classes *OLEBatchProcessJobDetailsController.java* and *OLEBatchProcessDefinitionController.java* extends the *TransactionalDocumentControllerBase* of Rice and overrides and adds necessary methods.

Once the Batch Process job is saved, the *BatchProcessFactory* checks for the Batch Process Type and based on that redirects to classes as shown in the below diagram.

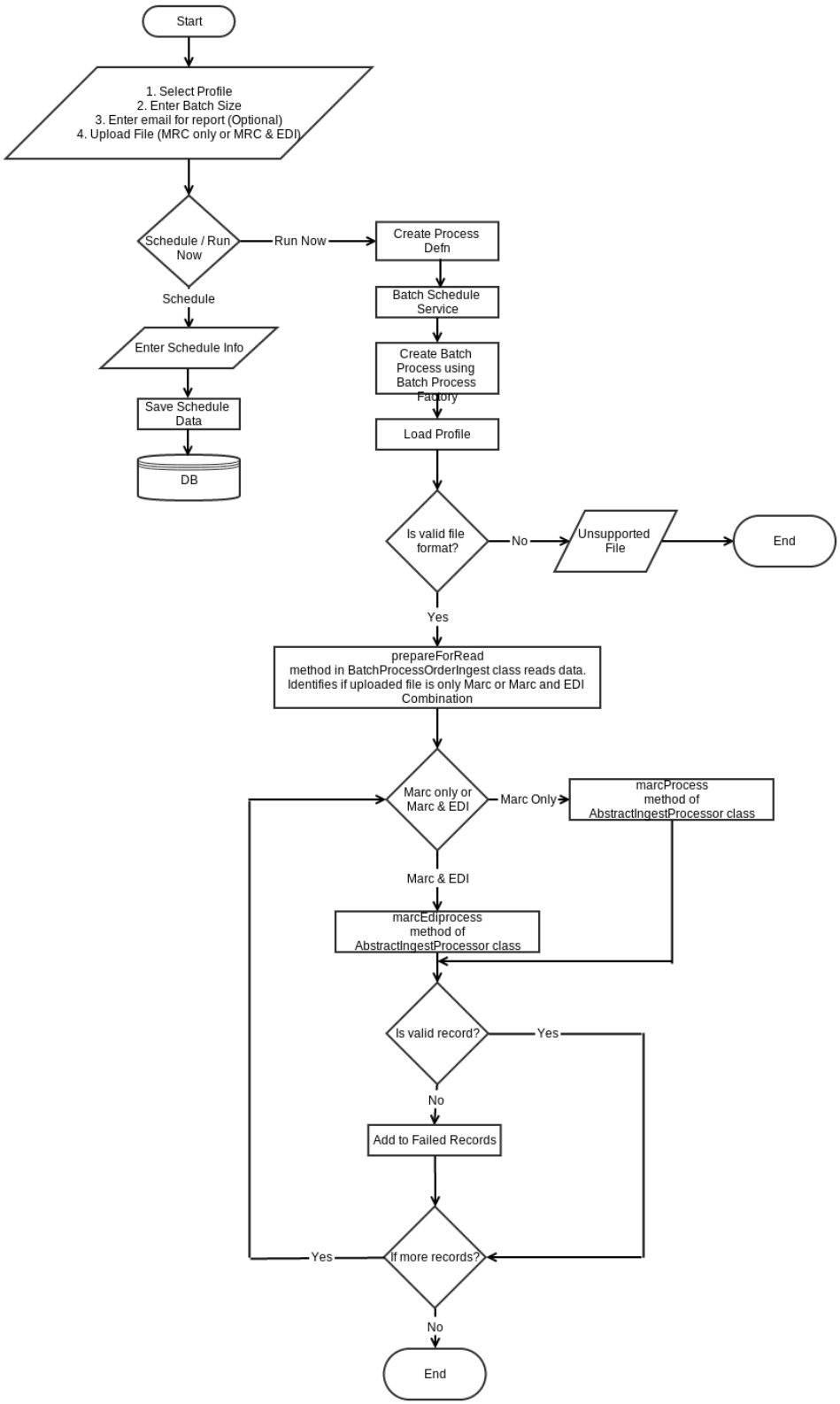


For more information, Javadocs can be found here.

Last Published December 8, 2015

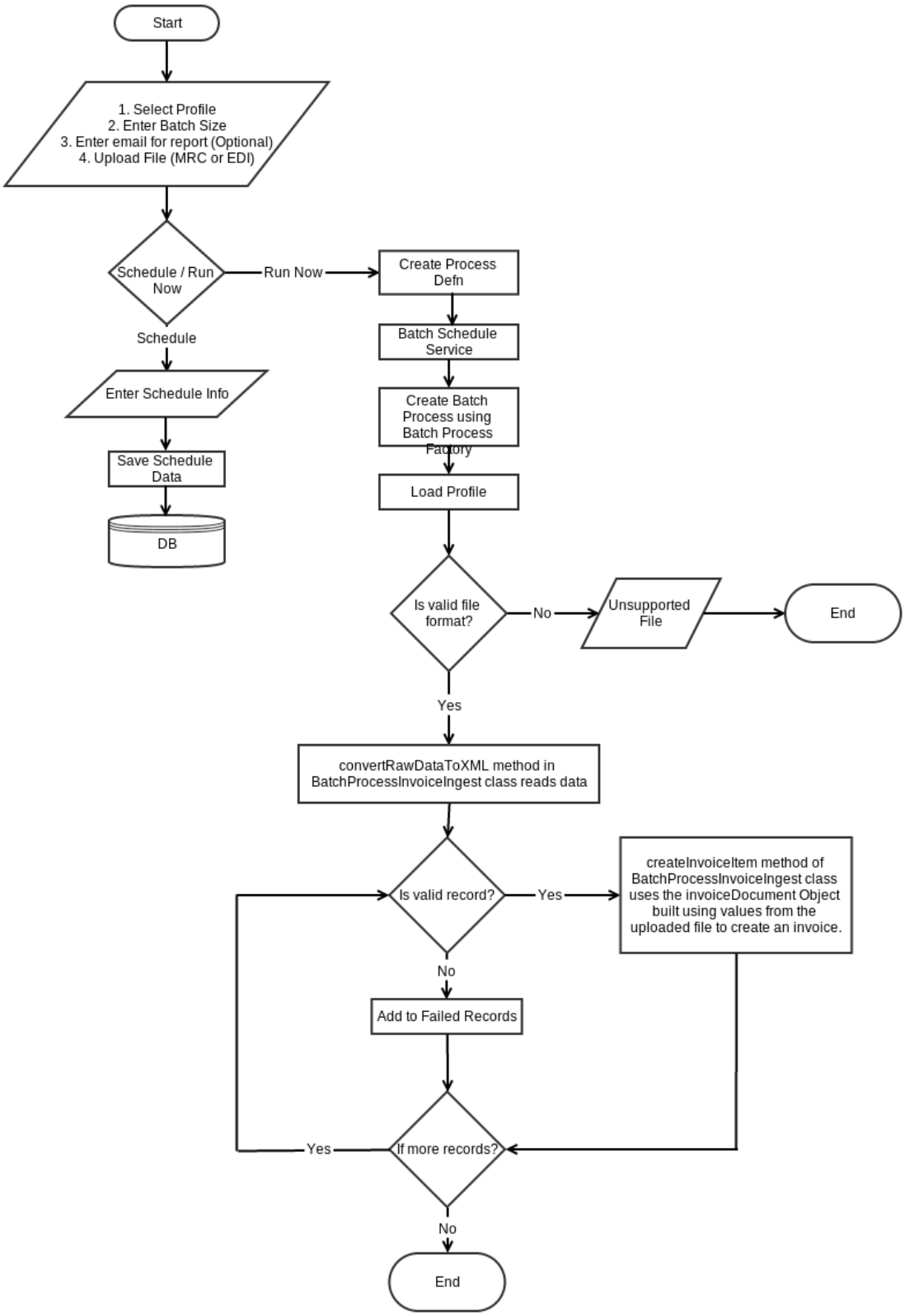
Flowchart

Order Record Import



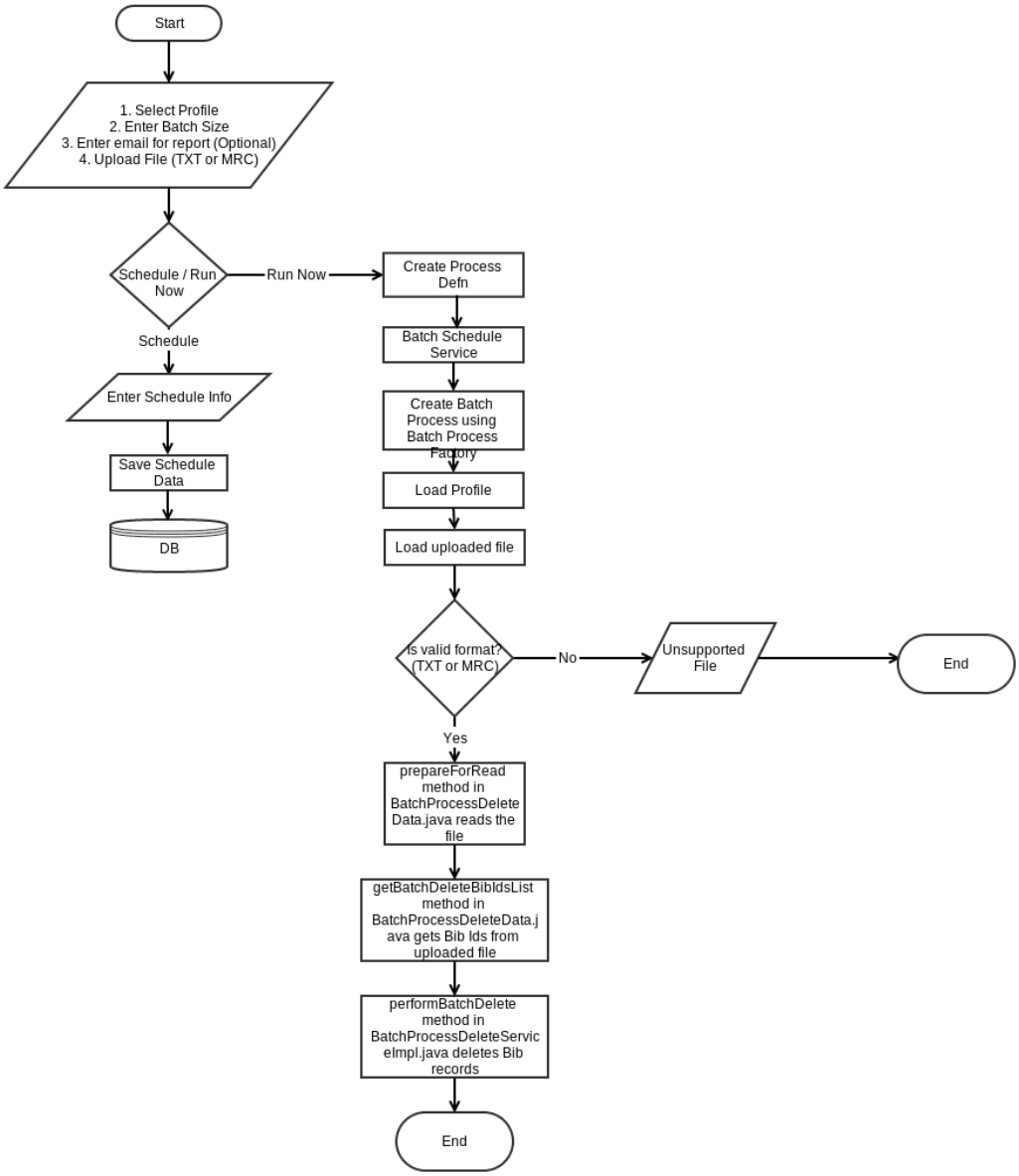
Last Published December 8, 2015

Invoice Import



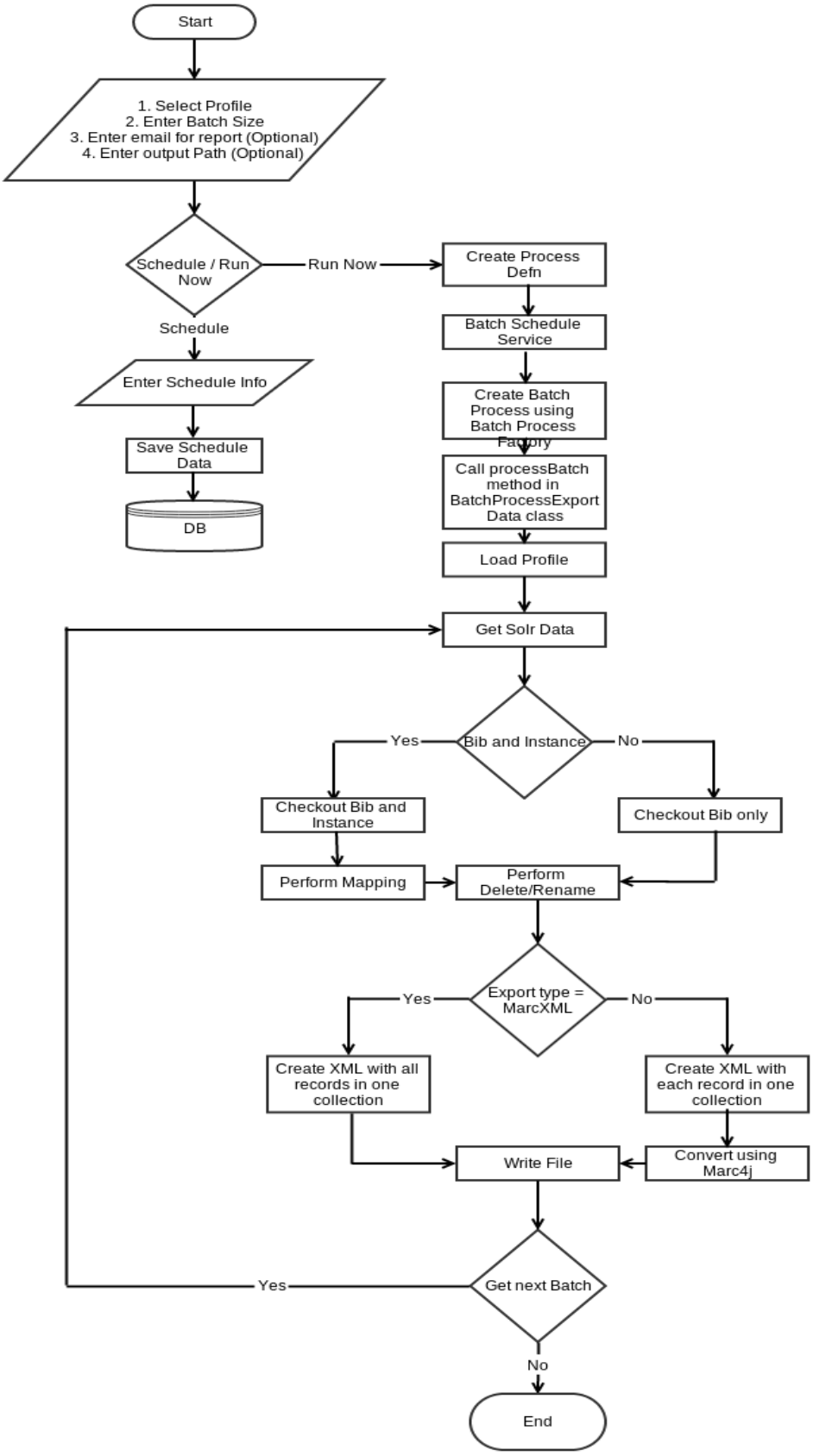
Last Published December 8, 2015

Batch Delete



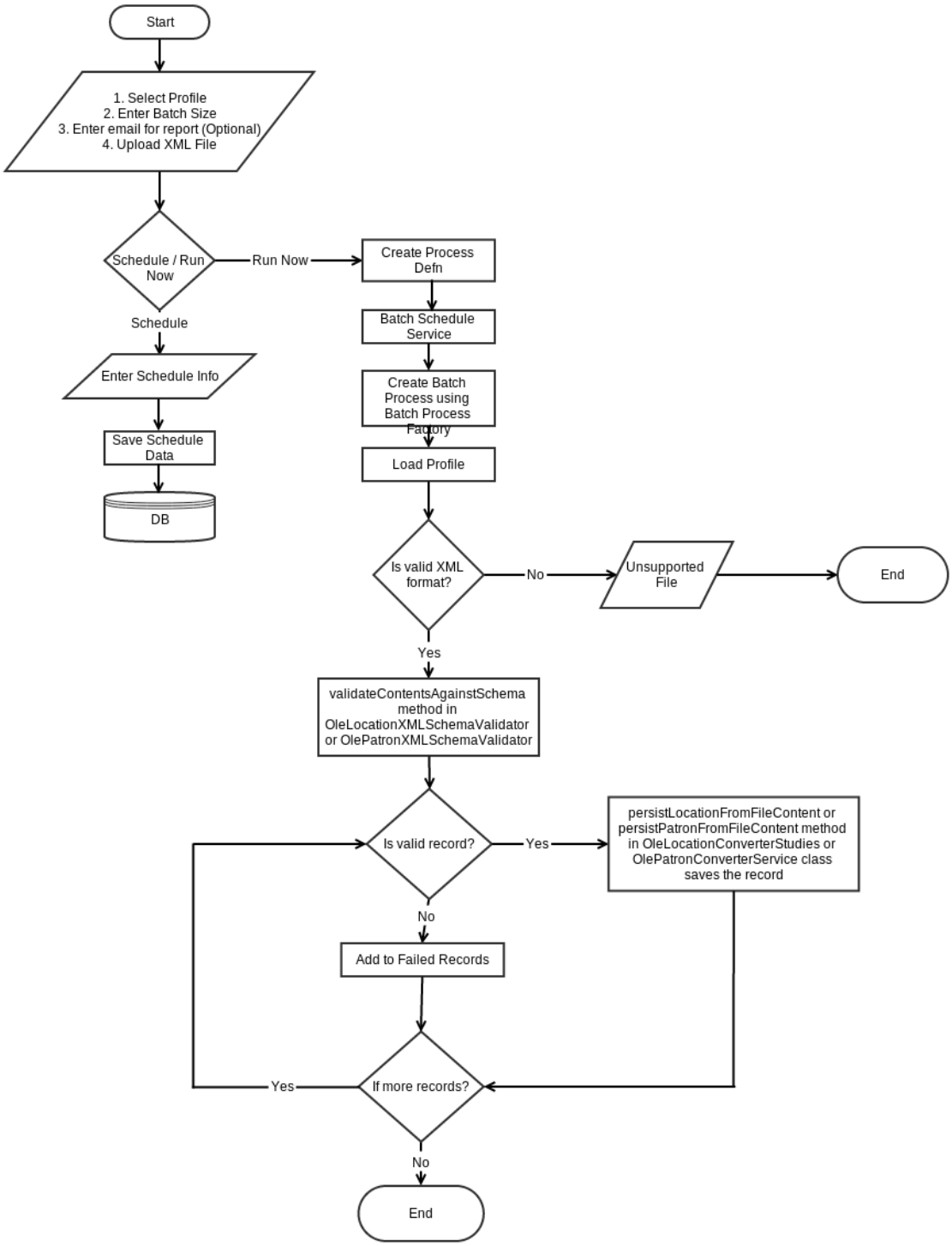
Last Published December 8, 2015

Batch Export



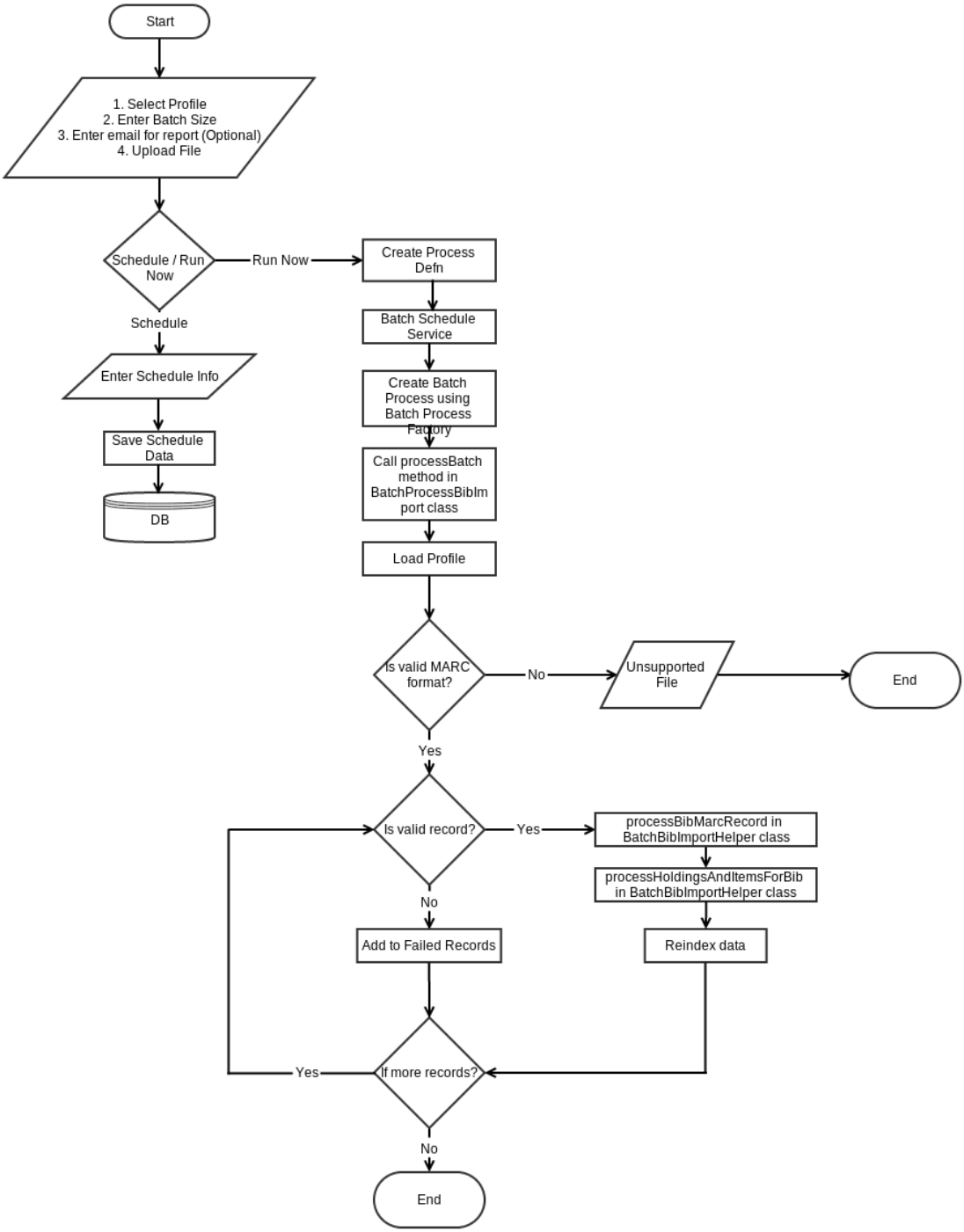
Last Published December 8, 2015

Patron / Location Import



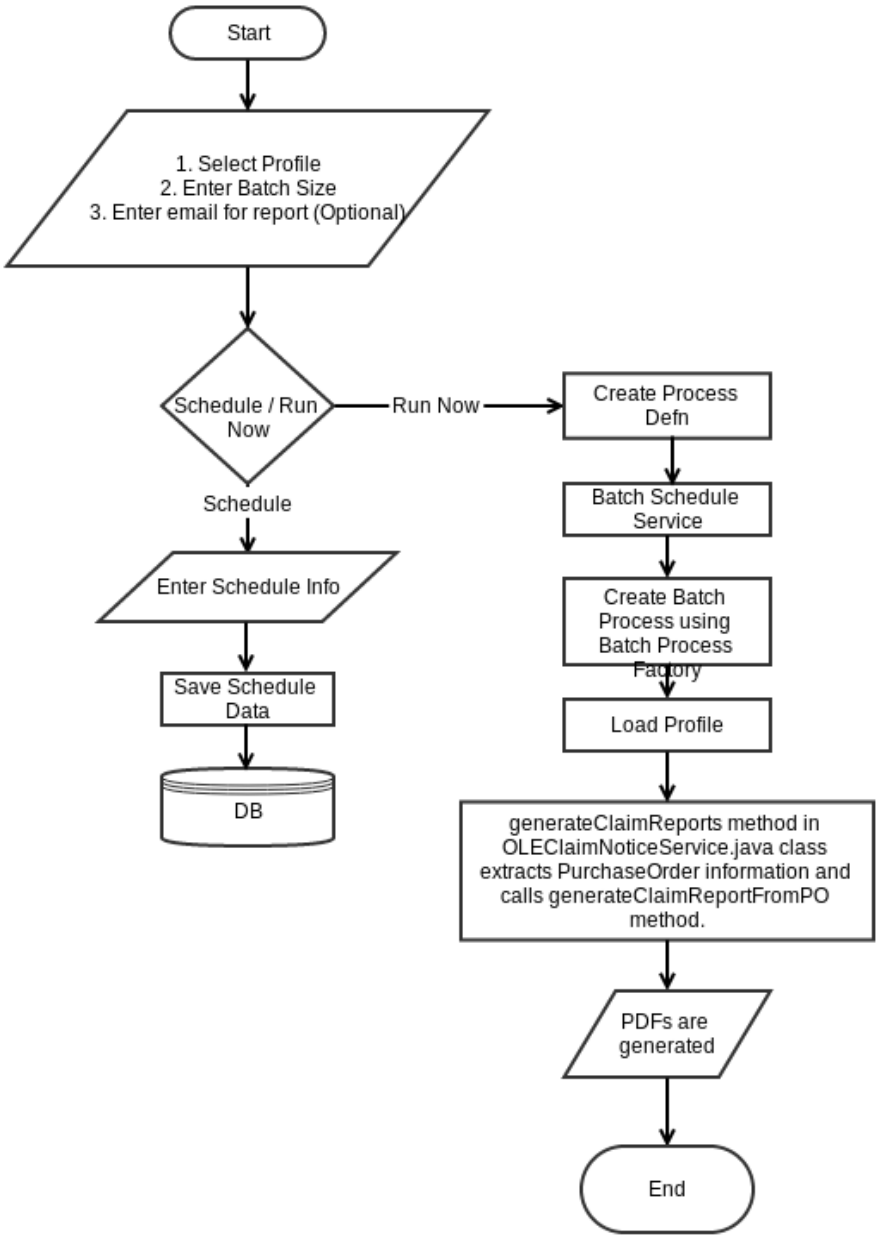
Last Published December 8, 2015

Bib Import

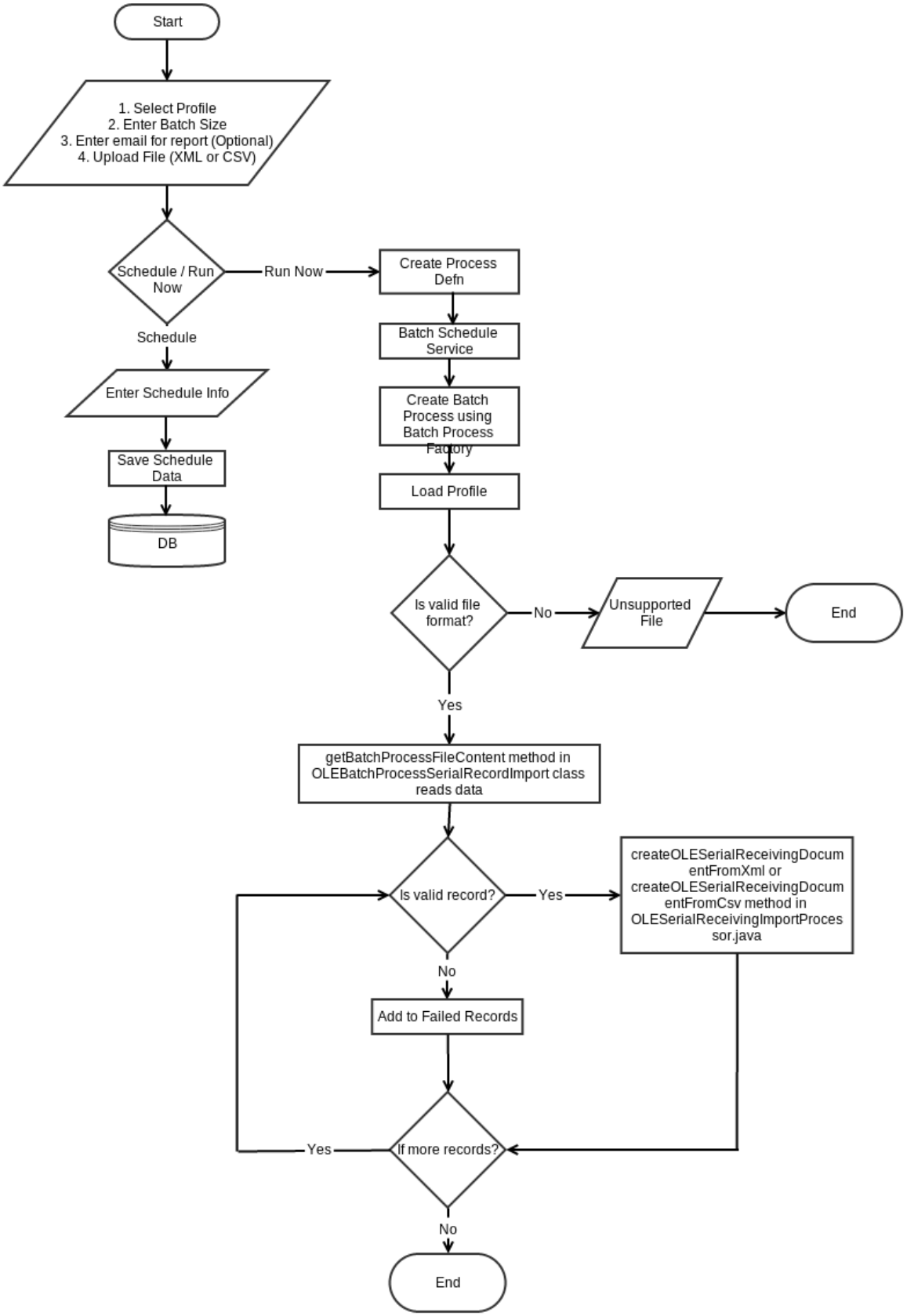


Last Published December 8, 2015

Claim Report



Serial Record Import



Last Published December 8, 2015

Pseudo Code

- User creates a Batch Process from the Batch Process Screen with Batch Process Name, Batch Process Profile, Batch Process Type. Based on the type, additional fields to enter data and upload files are shown.
- The user gets to schedule the Batch Process to happen according to a cron expression or set a particular time, either one time or recurring.

Order Record Import

- The Batch Process screen has an option to upload a MARC file or a combination of MARC and EDI files.
- The *getBatchProcessFileContent* method reads the uploaded file contents.
- Depending on the type of file(s) uploaded, either *marcEdiprocess* or *marcProcess* methods of AbstractIngestProcessor class is used.
- Data from the uploaded file is used to create Plain Old Java Objects (POJO) of order and bib record. With these the data gets ingested into the database.
- When multiple records are encountered, this process is repeated multiple times. If there is a failure, the record is captured and reflected in the failure data file.

Invoice Import

- The Batch Process screen for Invoice Import has an option to upload either a MARC file or an EDI file.
- Depending on the file format, *convertRawDataToXML* method in BatchProcessInvoiceIngest class uses *byPassLogicForPreProcess* method and *convertToXml* method in OLEINVConverter class for MARC and EDI respectively.
- POJOs are created by the method *convertXMLToPojos*(Map) and *convertXMLToPojos*(Map, String) for EDI and MARC files respectively. These POJOs are used to ingest the data.
- Failure records are captured in a failure data file.

Batch Delete

- The Batch Process screen for Batch Delete accepts either a MARC file or a txt file.
- The method, *getBatchDeleteBibIdsList* in BatchProcessDeleteData class, reads the uploaded file based on the file type and forms a List of Bib Ids from the file.
- The method, *performBatchDelete* in BatchProcessDeleteServiceImpl class, does the delete operation based on the Bib Ids from the List.
- Failure records are captured in a failure data file.

Batch Export

- The Batch Process screen for Batch Export takes in extra data as to what format the exported file should be in - MARC or MarcXML, maximum number of record in each file and output file name.
- The *performSolrQuery* method in BatchProcessExportData class creates the Solr query based on filter criteria and executes to retrieve the results.

- The *batchProcessExportFetch* method in BatchProcessExportData class calls the *call* method in BatchExportFetch class where the exported records are made into files.
- The *buildBibMarcRecords* method in BatchExportFetch class is called internally in the call method which builds the MARC records.
- The *processBatch* method in BatchProcessExportData class, depending upon the filetype specified writes the exported record into files.
- An error file is created when an error is encountered.

Logic behind File Name

- The export file name is dependent on the name specified in the Output file name while initiating the Batch Process document. If no name is specified the Batch process name is used.
- The file name follows the following pattern, *<file name>-<yyyy-mm-dd*T*hhmm>.* The *<file name>* is either the name specified as Output file name in the Batch process document or the Batch process name.
- Depending on the Batch size, the files are split. If there are more records than the Batch size then a new file is created and the name follows the following pattern, *<file name>-<yyyy-mm-ddThhmm>*-part*<file number>*. For example, if the Bib records to be exported is 100 and the Batch size is 10 then 10 files will be generated each with 10 records with part1, part2, etc. appended in the file name.

Patron Import

- The Batch Process screen for Patron Import allows uploading of a XML file.
- The *getBatchProcessFileContent* method in BatchProcessPatronIngest class reads the data from the uploaded file.
- The *validateContentsAgainstSchema* method in OlePatronXMLSchemaValidator class validates the contents of the XML.
- The *persistPatronFromFileContent* method in OlePatronConverterService class loads the data from the XML into the database.
- A rejection list is formed with Patron data that fails insertion and a failure file is created.
- Patron imports can be used to update patron records, matching is done on the Patron Id.
- Updates are completed as followed:
    - **Scenario 1:**
    - Existing Patron Address
        - 1) Home
        - 2) Campus
    - Update Patron Input
        - 1) Home
        - 2) Others
    - Output
        - 1) Home
        - 2) Campus
        - 3) Others
    - **Scenario 2:**
    - Existing Patron Address
        - 1) Home
        - 2) Home

        - 3) Home
        - 4) Campus
    - Update Patron Input
        - 1) Home
        - 2) Others
    - Output
        - 1) Home
        - 2) Campus
        - 3) Others

    If the Patron has 3 existing home address but update comes with one home address, OLE will remove the existing 3 home address and add the one new home address which is in the incoming update patron information.

Bib Import

- The Batch Process screen for Bib Import allows uploading of MARC file.
- The *prepareForRead* method in BatchProcessBibImport class calls the *getBatchProcessFileContent* method to read the uploaded file content.
- The *preProcessMarc* method in BatchProcessBibImportServiceImpl class helps in creating the XML content from the MARC file.
- The *saveBatch* method in BatchProcessBibImportServiceImpl class calls methods to create, update and delete Bibs, Holdings and Items. It also makes a list of mismatched records.
- The *processBibMarcRecord* and *processHoldingsAndItemsForBib* methods in BatchBibImportHelper class processes the data and loads them.
- The methods also append failure records as they encounter them along with reasons.

Matching Process

- The *processBibMarcRecord* also carries out the matching process. Here the Match point information is retrieved from the Batch Process Profile. The holdings, item and eholdings details of the bib are also retrieved.
- First a check is made to verify if the new bib is expected to match with an existing bib. If false then a new bib structure is created from the record and added to the existing records.
- If true, then *findMatchingBib* method is used to identify the matching bib by passing the profile and the bib record. The method identifies the matching bib by doing a search based on the field value and retrieving the bib using the bib id returned from the search.
- If no matching bib is found, then it is checked whether the new incoming bib is to be added or discarded and based on the value it is added or discarded.
- If matching bib is found, then it is checked whether the bib is to be added or updated.
- In case of just addition, the bib tree is built and added to the collection.
- In case of updating the bib, it is verified whether to overlay the bib. The *overlayBib* method takes care of deleting fields, renaming fields, setting defaults or constants, processing the 001 and overlaying other data fields.
- The *processHoldingsAndItemsForBib* method similarly checks for profile information on overlay and acts upon holdings, eholdings and item information. Here a check is made to verify if matching is to performed or not.

- If the profile says no matching is to be done, then the holdings record can be processed in three different ways - All incoming holdings and items are discarded, Existing holdings and items are discarded and incoming holdings and items are added or existing holdings and items are kept and incoming ones are added.
- If the profile says matching is to be done then *getMatchedHoldings* method is used to identify matching holdings.
- If no incoming holding matches existing holdings, based on the profile the incoming holdings and items are discarded or added. In case of adding the holdings, adding items is optional.
- If the incoming holding matches existing holdings, based again on the profile, the incoming holdings are either discarded or overlayed. All this happens in the *processMatchedPHoldings* method for print holdings and *processMatchedEHoldings* method for electronic holdings.
- In case of print holdings while being overlayed, items are to be processed. This happens in the *processItem* method.
- As in case of holdings, for items too the first check is whether matching is to be performed. If no matching is to be done, based on the profile three ways in which items are processed - all incoming items are discarded, existing items are discarded and incoming items added or existing items are kept and incoming items are added.
- If the matching is to be done and if no incoming item matches the existing items, depending on profile, incoming items are either discarded or added.
- If there is a match, items are either added or overlayed.
- While being added, if there are multiple holdings and multiple items, the mapping is done based on the presence of holding details like call number, call number prefix, call number type, copy number and holding location levels in item data. This is matched with the data on Holding record and the item linked to the holding. The *buildItemForHoldings* method of BatchBibImportHelper class is where this happens.

Location Import

- The Batch Process screen for Location Import allows uploading of a XML file.
- The *getBatchProcessFileContent* method in AbstractBatchProcess class reads the data from the uploaded file.
- The *validateContentsAgainstSchema* method in OleLocationXMLSchemaValidator class validates the contents of the XML.
- The *persistLocationFromFileContent* method in OleLocationConverterService class loads the data from the XML into the database.
- A rejection list is formed with Location data that fails insertion and a failure file is created in the createBatchFailureFile method in BatchProcessLocationIngest class.

Claim Report

- The *processBatch* method in OLEBatchProcessClaimReport class calls the *generateClaimReports* method of OLEClaimNoticeService class.
- A list of all Purchase Orders (POs) are sourced from the BusinessObjectService's *findAll* method.
- The *generateClaimReportFromPO* method in OLEClaimNoticeService class is called with List of POs. This in turn calls the *generatePdf* method with Vendor information in the

OLEClaimNoticeService class.

Serial Record Import

- The Batch Process screen for Serial Record Import allows uploading either a XML or CSV file.
- The *getBatchProcessFileContent* method in OLEBatchProcessSerialRecordImport class reads the file content.
- In case of XML, *createOLESerialReceivingDocumentFromXml* method in OLESerialReceivingImportProcessor class and in the case of CSV, *createOLESerialReceivingDocumentFromCsv* method in OLESerialReceivingImportProcessor class are used to extract data from the files.
- The *validateSerialReceivingDocument* method in OLESerialReceivingImportProcessor class validates the data.
- A failure data file with failed records is also compiled.

Sample XML

Patron

The below XML sample contains one Patron record, one can have multiple such Patron records within the <patronGroup> element.

```
<?xml version="1.0"?>
<patronGroup xmlns="http://ole.kuali.org/standards/ole-patron">
	<patron xmlns="http://ole.kuali.org/standards/ole-patron">
	<patronID>26821406</patronID>
	<expirationDate>2014-05-31</expirationDate>
	<activationDate>2013-09-25</activationDate>
	<active>true</active>
	<name>
		<first>Carrie</first>
		<middle>T</middle>
		<surname>Weston</surname>
	</name>
	<borrowerType>UGRAD</borrowerType>
	<barcode>26821406</barcode>
	<postalAddresses>
		<postalAddress>
		<postalAddressType>HM</postalAddressType>
		<addressLine>999 N Monticello</addressLine>
		<city>Chicago</city>
		<stateProvince>IL</stateProvince>
		<addressSource>REGL</addressSource>
		<postalCode>60647</postalCode>
		<default>true</default>
		<active>true</active>
		<deliverAddress>true</deliverAddress>
		</postalAddress>
		<postalAddress>
```

```
            <postalAddressType>CMP</postalAddressType>
            <addressLine>99 N. MONTICELLO</addressLine>
            <addressLine/>
            <city>CHICAGO</city>
            <stateProvince>IL</stateProvince>
            <addressSource>REGL</addressSource>
            <postalCode>60647</postalCode>
            <default>false</default>
            <active>true</active>
            <deliverAddress>false</deliverAddress>
            </postalAddress>
        </postalAddresses>
        <emailAddresses>
            <emailAddress>
            <emailAddressType>HM</emailAddressType>
            <emailAddress>sample55@uchicago.edu</emailAddress>
            <default>true</default>
            <active>true</active>
            </emailAddress>
        </emailAddresses>
        <telephoneNumbers>
            <telephoneNumber>
            <telephoneNumberType>HM</telephoneNumberType>
            <telephoneNumber>7737269999</telephoneNumber>
            <default>true</default>
            <active>true</active>
            </telephoneNumber>
            <telephoneNumber>
            <telephoneNumberType>CMP</telephoneNumberType>
            <telephoneNumber>7737269999</telephoneNumber>
            <default>false</default>
            <active>true</active>
            </telephoneNumber>
        </telephoneNumbers>
        <patronLevelPolicies>
            <isGenerallyBlocked>false</isGenerallyBlocked>
            <hasDeliveryPrivilege>true</hasDeliveryPrivilege>
            <hasPagingPrivilege>true</hasPagingPrivilege>
            <receivesCourtesyNotice>true</receivesCourtesyNotice>
        </patronLevelPolicies>
        <notes>
            <note>
            <noteType>GENERAL</noteType>
            <note>Student Division: COL</note>
            <active>true</active>
            </note>
        </notes>
        <localIdentifications>
            <localIdentification>
            <localId>1</localId>
            </localIdentification>
            <localIdentification>
            <localId>2</localId>
            </localIdentification>
        </localIdentifications>
```

```
      </patron>
</patronGroup>
```

Location

Similar to Patron, the Location XML can take in multiple locations under the <locationGroup> element.

```
<?xml version="1.0" encoding="UTF-8"?>
<locationGroup xmlns="http://ole.kuali.org/standards/ole-location">
    <location>
        <locationCode>B-AAAMC</locationCode>
        <locationName>Blmgtn - Arch. of African American Music &amp;amp; Culture</locationName>
        <locationLevelCode>LIBRARY</locationLevelCode>
        <parentLocationCode/>
    </location>
    <location>
        <locationCode>B-ALF</locationCode>
        <locationName>Blmgtn - Auxiliary Library Facility</locationName>
        <locationLevelCode>LIBRARY</locationLevelCode>
        <parentLocationCode/>
    </location>
<locationGroup>
```

Order

An Order import would involve uploading two files, a MARC and an EDI file. The MARC file contains the item information and EDI file contains the order information.

EDI Sample

```
UNA:+.? '
UNB+UNOC:3+YANKEE BOOKS:ZZ+DUL:ZZ+120306:1434+8'
UNH+8+ORDERS:D:96A:UN:EAN008'
BGM+220+8+9'
DTM+137:20120306:102'
NAD+BY+DUL::92'
NAD+SU+YANKEE BOOKS::92'
CUX+2:USD:9'
LIN+1'
PIA+5+9781589835542:IB'
QTY+21:1'
PRI+AAB:38.95'
RFF+SLI:99946641049'
RFF+BFN:DXXEETHREXXX631-2012'
FTX+LIN+++rll/ts'
NAD+OB+++TS'
LIN+2'
PIA+5+9780521193757:IB'
QTY+21:1'
PRI+AAB:199.00'
RFF+SLI:99946649868'
RFF+BFN:PXXUXXXBOXXX631-2012'
FTX+LIN+++egg/jmb'
NAD+OB+++EGG'
LIN+3'
PIA+5+9781845936884:IB'
QTY+21:1'
PRI+AAB:75.00'
RFF+SLI:99946649891'
RFF+BFN:PXXUXXXBOXXX631-2012'
FTX+LIN+++egg/jmb'
```

Last Published December 8, 2015

```
NAD+OB+++EGG'
LIN+4'
PIA+5+9780061906107:IB'
QTY+21:1'
PRI+AAB:27.99'
RFF+SLI:99946649991'
RFF+BFN:PXXUXXXSCXXX631-2012'
FTX+LIN+++egg/jmb'
NAD+OB+++EGG'
LIN+5'
PIA+5+9780321786678:IB'
QTY+21:1'
PRI+AAB:228.73'
RFF+SLI:99946650033'
RFF+BFN:PXXUXXXZOXXX631-2012'
FTX+LIN+++egg/jmb'
NAD+OB+++EGG'
UNS+S'
CNT+2:5'
UNT+49+8'
UNZ+1+8'
```

MARC Sample

A MARC file when opened with a third party tool like the MarcEdit looks like below. The MARC file is used in Bib import / delete operations and Invoice import and along with the EDI file in Order import.

```
=LDR  00407nam  2200145z  4500
=008  110630n\\\\\\\\xx\|\\\\\\\\\u|||\||und\u
=020  \\$a9780748636358
=100  \\$aWHONG, MELINDA
=245  \\$aLANGUAGE TEACHING : LINGUISTIC THEORY IN PRACTICE.
=260  \\$aEDINBURGH$bEDINBURGH UNIV PRESS$c2011
=960  \\$a65
=980  \\$b32.00$q1
=982  \\$a99943744781$b524010$dPaper$fCM
=984  \\$a20110629$cUS
=987  \\$a100552240
```

Service Interface Design (SOAP/REST)

No Web services are available currently.

User Interface Design

The Batch Process document uses KRAD's UIF (User Interface Framework). A very good guide on the framework can be found here. The Spring Beans XML used in the case of Batch Process are the *OLEBatchProcessDefinitionDocument.xml* under the *datadictionary* folder and *OLEBatchProcessDefinitionView.xml* under the *uif* folder.

Data Importing

It is currently not possible to import a Batch Process directly into OLE.

Data Exporting (if applicable)

OLE uses a RDBMS backend and hence any data can be exported using simple SQL queries.

Workflow

Batch Process document doesn't have any workflow defined by default. However, if needed it can be defined at *OleBatchDocType.xml* file.

System Parameters

Namespace Code	Parameter Name	Description
OLE-SYS	MAX_FILE_SIZE_UPLOAD	The size limitation for batch file uploads in megabytes.

Roles and Permissions

Permissions are linked to roles which are in turn linked to Users to give them access to screens and functions.

Permission ID	Permission Name
OLE1500	Upload Batch Input File(s)
OLE10243	BATCH_PROCESS_IMPORT
OLE10244	BATCH_PROCESS_EXPORT
OLE10271	BATCH_PROCESS_DELETE

Role ID	Role Name	Permissions
OLE10003	OLE_Load	OLE1500
OLE54	User	OLE1500
OLE10077	Cataloging Super User	OLE10243, OLE10244, OLE10271
OLE10078	Cataloger Supervisor	OLE10243, OLE10244, OLE10271
OLE10082	Full Cataloging	OLE10243, OLE10244, OLE10271
OLE10083	Batch Cataloging	OLE10243, OLE10244, OLE10271

Last Published December 8, 2015