Oracle® Revenue Management and Billing

Version 2.7.0.1.0

File Upload Interface Implementer's Guide

Revision 2.0

F25013-01

November, 2019

ORACLE

Oracle Revenue Management and Billing File Upload Interface Implementer's Guide

F25013-01

Hazardous Applications Notice

This software is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure the safe use of this software. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software in dangerous applications.

Third Party Content, Products, and Services Disclaimer

This software and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third party content, products, or services.

Preface

About This Document

This document provides a detailed explanation of ORMB approach for Data Conversion and Integration. It describes parameters related to File Upload Interface Master Configuration and explains how to perform important tasks using File Upload Interface. This Guide supplements the information provided in *File Upload Interface User Guide* and *File Upload Interface Batch Execution Guide*.

Intended Audience

This document is intended for the following audience:

- End-Users
- Development Team
- Consulting Team
- Implementation Team

Organization of the Document

The information in this document is organized into the following sections:

Section No.	Section Name	Description
Section 1	ORMB Approach for Data Conversion on Premise	Explains the ORMB Approach for Data Conversion on premise. It also provides the steps to be followed for converting data using ORMB.
Section 2	File Upload Interface	Provides an overview of File Upload Interface
Section 3	ORMB Approach for Data Conversion using FUI	Explains the ORMB Approach for Data Conversion using File Upload Interface. It also provides the steps to be followed for converting data using File Upload Interface.
Section 4	File Upload Interface Master Configuration	Describes important parameters related to File Upload Interface Master Configuration
Section 5	Creating File Request Type	Lists the steps to create file request type
Section 6	Working with File Request Type	Explains different fields in File Request Type zone and tasks which you can perform using File Upload Interface
Section 7	Updating records marked with 'Error' or 'Pending' status	Lists steps to update record status using File Upload Dashboard
Section 8	Transforming Data	Identifies important concepts related to transforming data

Section No.	Section Name	Description
Section 9	File Management System	List of files that are ready to be uploaded in ORMB system
Section 10	Rule Configuration for SFTP Poller File Upload Batch	Rule engine details used in SFTP Poller batch for file polling and uploading in ORMB system

Related Documents

You can refer to the following documents for more information:

Document	Description
Oracle Revenue Management and Billing Banking User Guide	Lists and describes various banking features in Oracle Revenue Management and Billing. It also describes all screens related to these features and explains how to perform various tasks in the application.
File Upload Interface User Guide	Helps you configure File Upload Interface.
File Upload Interface Batch Execution Guide	Explains the batches to be executed while performing various tasks in File Upload Interface.

Conventions

The following conventions are used across the document:

Convention	Meaning
boldface	Boldface indicates graphical user interface elements associated with an action, or terms defined in the text.
italic	Italic indicates a document or book title.
`monospace`	Monospace indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or information that an end-user needs to enter in the application.

Contents

1. ORMB Approach for Data Conversion on Premise

Enterprise information systems comprise of a variety of data storage systems, which vary in complexity and in the ways they access internal data.

- **Shared Database** - All applications that you are integrating read data directly from the same database.
- **Maintain Data Copies** - Maintain copies of the application's database so that other applications can read the data (and potentially update it).
- **File Transfer** - Make the data available by transporting a file that is an extract from the application's database so that other applications can load the data from the files.
- **Service Integration** - Real time integration using SOAP/REST services.

ORMB uses **File Transfer** approach for data conversion and data integration.

1.1 Prerequisites

To convert data using ORMB on premise, you should have:

1. Conversion tool kit
2. Pre-staging, staging and production schema
3. Stored procedures to map (transform) and transfer data from **pre-staging to staging** and **staging to production**

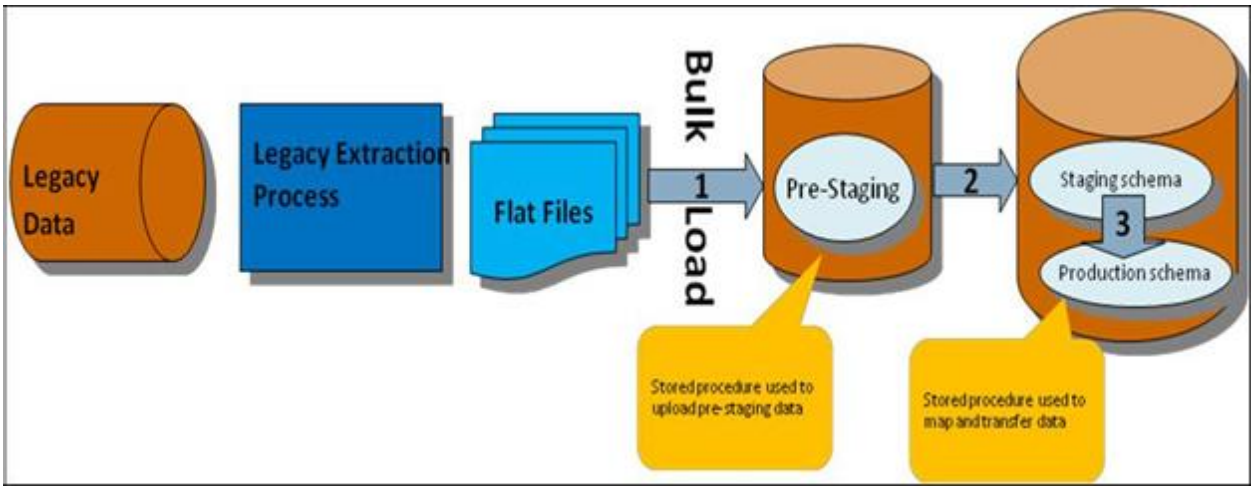
1.2 Steps for ORMB Conversion

To convert data, you need to follow the below steps:

1. The legacy data has to be manually relocated to a database for ConversionMapper to access the data before converting into ORMB table structure.
2. Create a Pre-Staging schema.
3. Define tables to refer to legacy tables external to ORMB. These tables (that contain legacy data) are defined as Input Tables in ConversionMapper.
4. A Bulk Load utility such as SQL*Loader in Oracle is typically used to copy legacy files into ORMB Pre-Staging schema.
5. Create a Staging schema.
6. Instead of using the flat files directly, Oracle tables (Input Tables) that represent the flat files are used to map to ORMB.
7. The legacy tables or “Input Tables” in ConversionMapper MUST exist on a database (in staging schema), so that ConversionMapper can use SQL statements to access the tables.
8. The legacy tables can be defined as tables on the staging schema (STGADM) or as views (also on STGADM) of tables on another schema or database.
9. Stored Procedures are used to map, validate and transfer the data to staging schema.

1.3 ORMB Conversion Flow Diagram

The flow of data between the two systems is illustrated below:



1.4 Disadvantages

- Data transformation and loading is done using Stored procedures thereby resulting in duplication of data validation and business rules.
- Risk of missing data validation and business rules.

1.5 Cloud Guideline

Before deciding to migrate to cloud based frameworks, note that:

- Legacy systems do not have access to ORMB database.
- Stored procedures cannot be deployed on cloud database.

2. Introduction to File Upload Interface

ORMB File Upload Interface is used for data conversion and data integration on Cloud. It provides ability to upload files to staging and map file records to ORMB services. ORMB File Upload Interface invokes ORMB services for each of the record. The ORMB file upload interface provides following benefits:

- Conversion tool kit not required

2.1 Pre-staging schema not required

- ORMB DB access not required for any legacy system
- Data load in ORMB is done using existing ORMB service schemas, thereby no duplication of data validation and business rules
- No risk of missing data validation and business rules
- Supports transformation for files in XML, JSON, Fixed Position, CSV, PSV and TSV formats
- Online system can be used to view the uploaded files and their corresponding processed details

3. ORMB Approach for Data Conversion and Integration using File Upload Interface

This chapter lists the steps to be followed before using File Upload Interface. It also describes approaches you can take when working with ORMB File Upload Interface.

3.1 Prerequisites

To convert data using ORMB using File Upload Interface, you should:

- **Define File Upload Interface master configuration**: For more information on how to define File Upload Interface master configuration, refer File Upload Interface User Guide.
- **Configure File Request Type** with required mapping of existing ORMB Services Business Objects or Business Services or Service Script specific to Transaction Stage Upload service. You need to add date format in `FILE_UPLOAD_DATE_FORMATS` lookup to support client-specific date inputs in Transaction Header.

3.2 Approaches for ORMB Conversion and Integration Process

To convert or integrate files in ORMB file upload interface, you can follow any of the below two approaches:

- ORMB conversion and integration process **without** Data Transformation
- ORMB conversion and integration process **with** Data Transformation

3.2.1 ORMB Conversion and Integration without Data Transformation

To convert or integrate files without Data Transformation, you need to follow below steps:

1. Ensure that the files are in XML format and comply with ORMB service schema.
2. Publish service XMLs conforming to ORMB service schemas that are to be used for conversion.
3. If validations for Header, Footer or Checksum are required, then you need to write an algorithm deriving `FileValidationAlgorithmSpot`.
4. If any preprocessing is required before invoking Business Object or Business Service or Service Script service, then you need to implement an algorithm deriving `FileRequestPreProcessingAlgorithmSpot`.
5. Execute File Transform and Upload batch (**C1-FTRAN**). This reads flat file records and upload in file upload staging. For more information on C1-FTRAN batch, refer to File Upload Interface Batch Execution Guide.
6. Execute File Request Processing batch (**C1-FREQP**). This reads the records from staging and process those using Services configured in file request type. For more information on C1- FREQP batch, refer File Upload Interface Batch Execution Guide.

7. File upload and processing steps can also be done by executing only C1-FTRAN batch. You are required to set “File Upload and Process” flag as true in File Request Type configuration.
8. Perform required fixes in File Request Type Configuration or File Details.

Note: During conversion process, conversion cycles can be run on UAT environment while the system is being tested on the production schema.

3.2.2 ORMB Conversion and Integration with Data Transformation

To convert or integrate files with Data Transformation:

File upload supports files in XML, JSON, CSV, PSV, TSV or Fixed Position formats.

1. Data Transformation algorithm can be implemented by deriving “FileRequestTranformationAlgorithmSpot”.
2. Data Transformation will be done using Data Transformation configuration in its file request type.
3. If validations for Header, Footer or Checksum are required, then implement an algorithm deriving “FileValidationAlgorithmSpot”.
4. If any preprocessing is required before invoking Business Object or Business Service or Service Script service, then implement an algorithm deriving “FileRequestPreProcessingAlgorithmSpot”.
5. Execute File Transform and Upload batch (C1-FTRAN). This will read the flat file records and transform those using “Data Transformation” algorithm and upload in file upload staging.
6. Execute File Request Processing batch (C1-FREQP). This will read the records from staging and process those using Service configured in File Request Type.
7. File transform, upload and processing steps can also be done by executing only C1-FTRAN batch. You are required to set “File Upload and Process” flag as true in File Request Type configuration.
8. Perform required fixes either in File Request Type Configuration or file details.

Note: During conversion process, conversion cycles can be run on UAT environment while the system is being tested on the production schema.

4. FUI Master Configuration

This configuration is referred to in file upload for file decryption, archival of file, and audit logging. To view and edit file upload configuration, you need to do the following:

1. From the Admin menu, select M and then click Master Configuration. The Master Configuration window appears.
2. Click Edit corresponding to the File Upload Interface Configuration. The Master Configuration window appears.

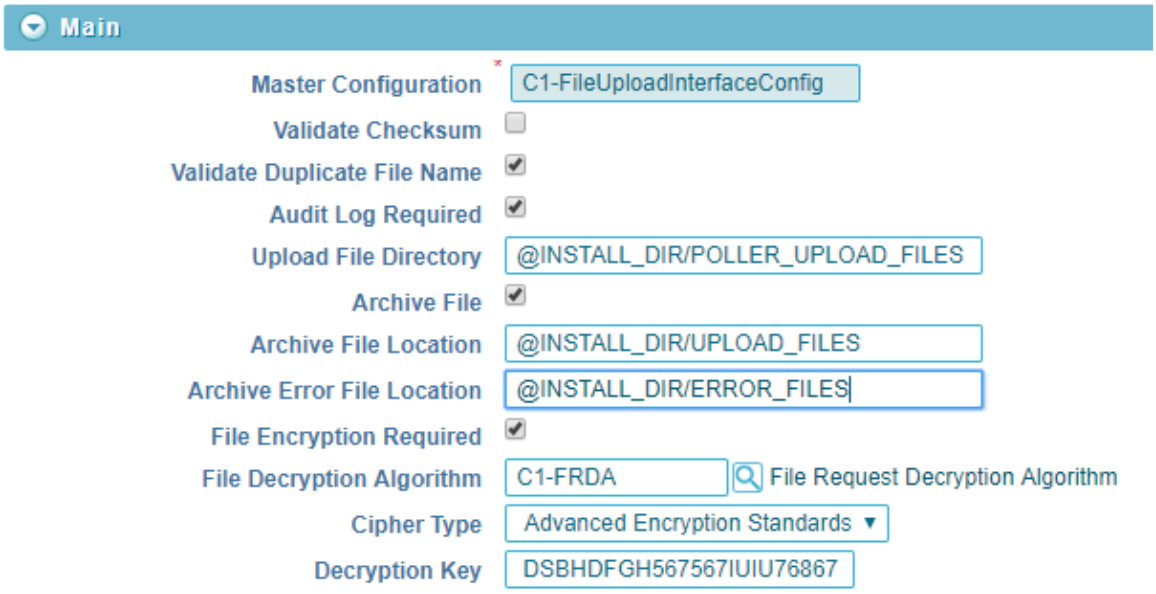


Figure 1: File Upload Interface Configuration - Master Configuration

4.1 Parameters related to FUI Master Configuration

This section lists and describes following important parameters related to File Upload Interface Master Configuration:

- Validate Checksum
- Validate Duplicate File Name
- Audit Log
- Archive File
- Archive File Location
- Archive Error File Location
- File Encryption
- File Decryption Algorithm
- Cipher Type
- Decryption Key

4.1.1 Validate Checksum

This flag decides whether to check the integrity of the file before staging file contents in ORMB system or not. The Validate Checksum parameter is a check box:

- If **selected**, checksum validation is performed for all the uploaded files
- If **not selected**, checksum validation is skipped

A few pointers regarding Checksum validation:

- Every uploaded file must have a corresponding `<FILE_NAME>.checksum` file on SFTP server.
- A File validation algorithm must be mapped with every File Request Type, which is derived from `FileHeaderValidationAlgorithmSpot`.
- The Checksum validation logic is implemented in the File validation algorithm, which is also used for Header and Footer validations.

File Validation algorithm is derived from "FileHeaderValidationAlgorithmSpot".

A sample algorithm `FileHeaderValidationAlgorithm_Impl` is available with:

- Header validation – Number of records are checked
- Checksum validation – Done using **MD5** algorithm type

For checksum validation, if you want to use the default implementation in any specific File validation algorithm, then it can be done by invoking `calculateChecksum(String fileString, String algoName)` function in `FileRequestProcessBusinessComponent_Impl` business component.

Note: For uploaded file, it is not required to have corresponding <FILE_NAME>.checksum file.

4.1.2 Validate Duplicate File Name

This flag is used to decide the validation of duplicate file name before uploading a file. The Validate Duplicate File Name parameter is a Check box with valid values True and False.

- If **True**, if a file with same name exists in staging, then the file is uploaded
- If **False**, file with same name is uploaded

4.1.3 Audit Log

This flag decides whether to log corresponding status changes of an individual file request after processing. The Audit Log parameter is a Check box with valid values True and False.

- If **True**, status transitions for all file requests are maintained in ORMB
- If **False**, status transitions for the file requests are not performed

Logging is done in `CI_FILE_REQUEST_DTL_MSG` table with `LOG_ENTRY_TYPE_FLG` value as **F1ST**. `CI_FILE_REQUEST_DTL_MSG` table is also used for error logging with `LOG_ENTRY_TYPE_FLG` value as **F1EX**.

4.1.4 Archive File

This flag decides whether to relocate the file to another location after processing on SFTP server. Here, location refers to the path mentioned in “Archive File Location” or “Archive Error File Location”. The Archive File parameter is a Check box with valid values True and False.

- If **True**, files are moved to another location. There are two scenarios:
    - If file is uploaded successfully, the files are moved to a defined archive file location.
    - If file upload fails, the files are moved to a defined archive error file location.
- If **False**, the files remain at the same location

4.1.5 Archive File Location

It is used to specify the file path used for archiving the file. The successfully processed file is moved to this location. The Archive File Location parameter is a field where you can enter the location.

- File Location should always be a combination of logical path and relative path and prefixed with either `INSTALL_DIR` or `SHARED_DIR`. For example, if you want to define file location as “/scratch/rmbbuild/sftpFile”, then it will be **INSTALL_DIR/ sftpFile** where `INSTALL_DIR` value is defined as “/scratch/rmbbuild”.
    - Define `INSTALL_DIR` variable value against `spl.runtime.environ.SPLEBASE` property in spl.properties file.
    - `SHARED_DIR` variable is a shared storage path mounted in cloud environment and has a static value.
- The defined location is appended by the corresponding File Request Type. For example, Archive Error File Location is **INSTALL_DIR/FilesUploaded**, where INSTALL_DIR path is **/scratch/rmbbuild**. If a batch is executed for `ADD_PERSON` file request type, then files are moved to **/scratch/rmbbuild/FilesUploaded/ADD_PERSON/**

4.1.6 Archive Error File Location

It is used to specify the file path used for archiving the error files. The files with errors will be moved to this location. The Archive Error File Location parameter should be a combination of logical path and relative path and prefixed with either `INSTALL_DIR` or `SHARED_DIR`. The defined location will be always appended by the corresponding File Request Type. For example, Archive Error File Location is **INSTALL_DIR/ErrorFiles** and INSTALL_DIR path is **/scratch/rmbbuild**. If a batch is executed for `ADD_PERSON` file request type, then files are moved to **/scratch/rmbbuild/ErrorFiles/ADD_PERSON/**

4.1.7 File Encryption

This flag decides whether to first decrypt and then extract the files on SFTP server. The File Encryption parameter is a Check box with values: True or False.

- If **True**, files on SFTP server are decrypted using File Decryption Algorithm, then extracted and processed to upload the file data in ORMB staging.
- If **False**, files on SFTP server are uploaded without decryption.

For using the file encryption parameter, a File Decryption Algorithm should be available. The algorithm is derived from `FileRequestDecryptionAlgorithmSpot` and has implementation commands required for getting the decrypted file.

A sample algorithm `FileRequestDecryptionAlgorithm_Impl` is provided with ORMB application and:

- Gets the decryption key for defined `com.oracle.ouaf.system.keystore.file` alias in **ouaf_keystore** file
- Gets the decrypted file using decryption key

4.1.8 File Decryption Algorithm

It defines the algorithm to be used for decryption of a file. You can search for the File Decryption Algorithm `FileRequestDecryptionAlgorithmSpot`. This algorithm must have an implementation for decrypting the input File.

- It gets the secret key stored against

  `com.oracle.ouaf.system.keystore.passwordFileName` alias and uses secret key to access ouaf_keystore file.
- It gets the decryption key stored against `com.oracle.ouaf.system.keystore.file` alias in ouaf_keystore file.
- The decryption key decrypts and returns the file string.

4.1.9 Cipher Type

This is an algorithm type used to get the decryption key. The Cipher Type parameter has a drop down list with values:

- Advanced Encryption Standards
- Data Encryption Standard
- RSA - RSA with AES

This parameter lists the decryption algorithm types.

4.1.10 Decryption Key

This key is used to decrypt the uploaded encrypted files on SFTP server. The Decryption Key parameter holds a decryption key, which is further used to get the decrypted file data. This field value is not stored in the database, but is stored against `com.oracle.ouaf.system.keystore.file` alias in ouaf_keystore file.

4.1.11 Upload File Directory

This Upload File Directory holds the root directory path of SFTP server where third party files can be uploaded. All the files in this directory and subdirectories are picked by SFTP File Upload Poller batch and uploaded in ORMB system using Rule Engine and DBMS scheduler batch job. This directory path is also used by File Management System UI to display all available files in the location.

5. Creating File Request Type

File request type is a configuration that allows you to upload files in any format and transform the files in ORMB compliant format.

When creating a file request type, you have two options:

- Create File Request Type without Data Transformation
- Create File Request Type with Data Transformation

5.1 Creating File Request Type without Data Transformation

To create file request type without Data Transformation:

1. Navigate to Admin > F > File Request Type > Add. The File Request Type window appears. This window has following sections:
   - Main
   - Services
   - Messages
   - Data Transformation
2. Enter name of file request type in File Request Type field.
3. Verify if the value for File Format field is Extensible Markup Language.

Note: Default value for File Format field is Extensible Markup Language.

- If the value is **Extensible Markup Language**, go to Step 5.
- If File Format has any other value,
   a. Select Data Transformation Required check box. The File Format field gets enabled.
   b. Select Extensible Markup Language from the File Format drop-down list.
   c. Deselect Data Transformation Required flag and go to Step 5.

Figure 2: Data Transformation

4. Configure at least one service within Service section. This service will be used to process flat file records.

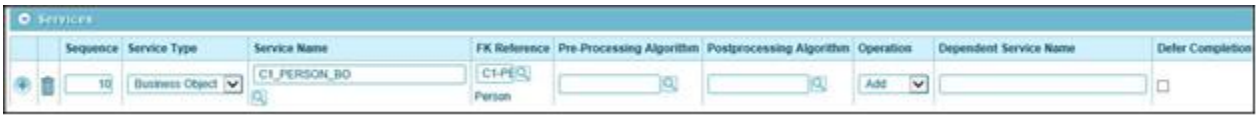


Figure 3: Configuring Service

5. Multiple relational or non-relational services can be configured under single File Request Type. For multiple services, service execution sequence of each record is decided on the basis of given Sequence number.

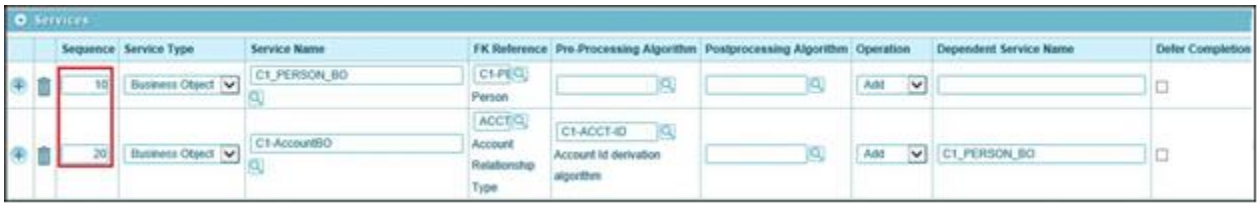


Figure 4: Configuring Multiple Services without Data Transformation

6. Click Save.

5.2 Creating File Request Type with Data Transformation

To create file request type with Data Transformation:

1. Navigate to Admin > F > File Request Type > Add. The File Request Type window appears.
2. Enter name of file request type in File Request Type field.
3. Select Data Transformation Required check box. The File Format field gets enabled.

Figure 5: Data Transformation

4. Select the required file format from File Format drop-down list. The valid values are:
   - Comma Separated Values (CSV)
   - Extensible Markup Language (XML)
   - Fixed Position
   - JavaScript Object Notation (JSON)
   - Pipe Separated Values (PSV)
   - Tilde Separated Values (TSV)

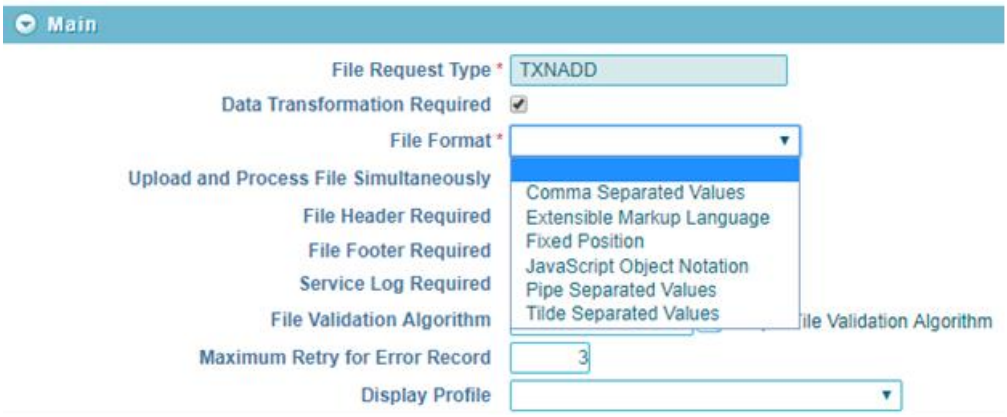


Figure 6: File Format

5. Configure at least one service within Service section. This service which will be used to process flat file records.

Figure 7: Service Sequence

6. Multiple relational or non-relational services can be configured under single File Request Type. For multiple services, service execution sequence of each record is decided on the basis of given Sequence number.

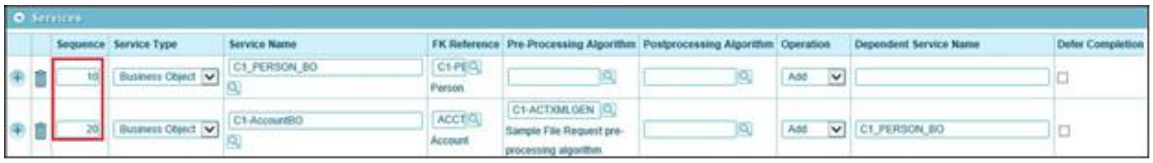


Figure 8: Configuring Multiple Services with Data Transformation

7. If data transformation required, then, Data Transformation details need to be configured with respect to its corresponding flat file data line. For more information on file data line, refer section Mapping Data Line or Record in a File.

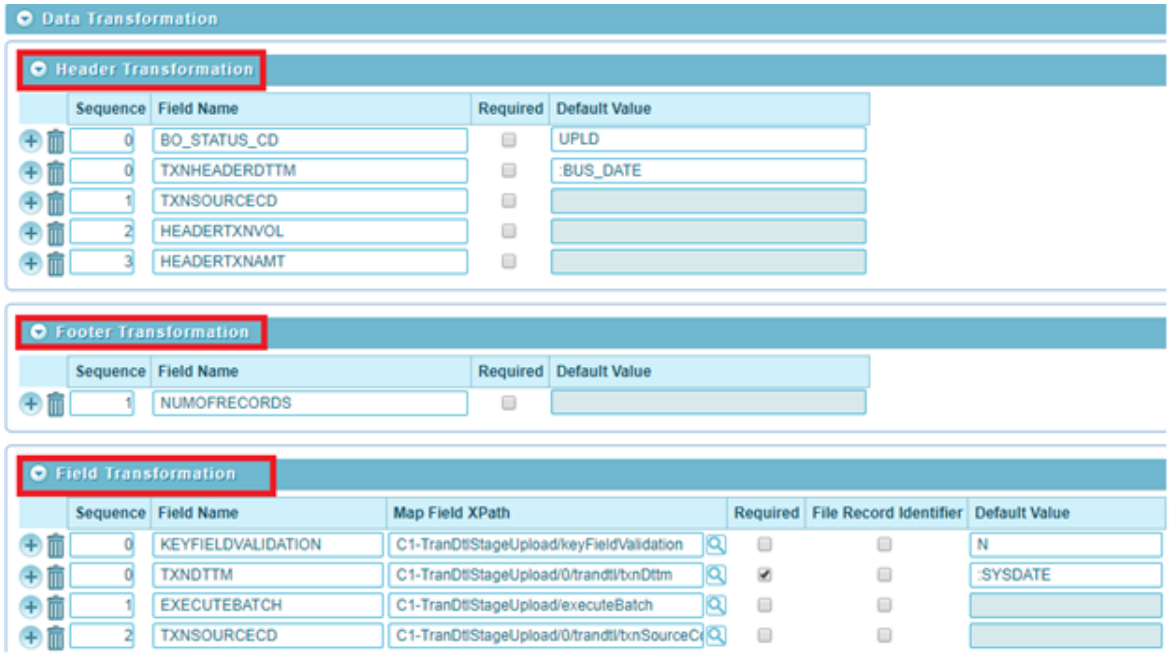


Figure 9: Configuring Data Transformation

8. Click Save.

6. Working with File Request Type

6.1 Uploading and Processing Records using Single Batch Execution

To upload and process records using single batch execution:

1. Define a new file request type or search for an existing file request type. Select **Upload and Process File Simultaneously** check box. This sets the flag as **True**.

Figure 10: Upload and Process Records

2. Execute **C1-FTRAN** batch. This batch uploads the file and start processing of all records in the file.

6.1.1 File Extensions used for Reading Uploaded Files on SFTP server

File Upload interface uses flat files to upload data into ORMB system. These flat files can have extensions like **.txt**, **.csv**, **.dat**, .**xml** etc. The legacy system locates these files on SFTP server and uploads data from it. It is possible that the legacy system locates different files with different extensions at the same location. Hence, the **C1-FTRAN** batch reads only those files that matches extensions configured in File Request Type.

Figure 11: File Extension

6.1.2 Rollback all Processed Records in case of Single Record Failure

In case of single record failure in a file, you can rollback all those processed records by executing **C1-FTRAN** batch.

To rollback processed records, select **File Atomicity** check box against the File Request Type. This automatically selects the **Upload and Process File Simultaneously** parameter.

The **C1-FTRAN** batch executes both upload and process in a single execution.

Figure 12: Rollback Processed Records

Note: Since, the batch is executed in single thread, it has a performance impact. Hence, File Atomicity should be opted only in case of low data volume.

Note: **C1-FTRAN** batch is executed using Single Transaction Strategy.

6.1.3 Validating Processing Details

The **Service Log Required** attribute validates if the processing details need to be captured for individual records or not. If you require service log, select **Service Log Required** check box against the File Request Type.

Figure 13: Service Log

Record details can have a primary key with its service name. Primary key for that record is stamped only if FK reference is configured in the File Request Type for the invoked service.

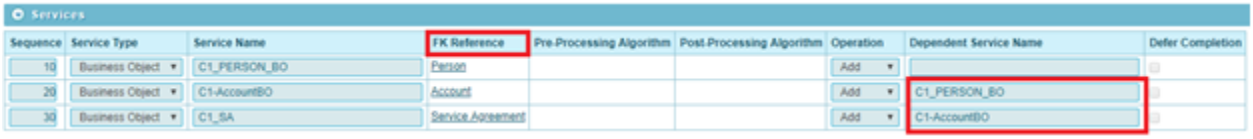


Figure 14: Service Log and Foreign Key Reference

These details in combination with foreign key reference is used to navigate to its respective entity like ‘Person’ or ‘Account’ or ‘Contract’, etc.

Note: This feature is Optional to optimize the performance of batch.

6.1.4 Header and Footer Details

Header and footer details are optional in a file. You can have both header and footer details in a file that you are uploading. If you want to upload a file with header and footer details, you need to configure Header and Footer details in the corresponding File Request Type.

Note: It is mandatory to implement **File Validation Algorithm** if the provided file has either of header or footer details. Header and footer details are used only for file validations.

- If **Data Transformation Required** check box is selected and the selected **File Format** is Extensible Markup Language, then:
  - **Root XML Tag** is mandatory. This means the service payload has root XML tag as **<request>……..</request>**

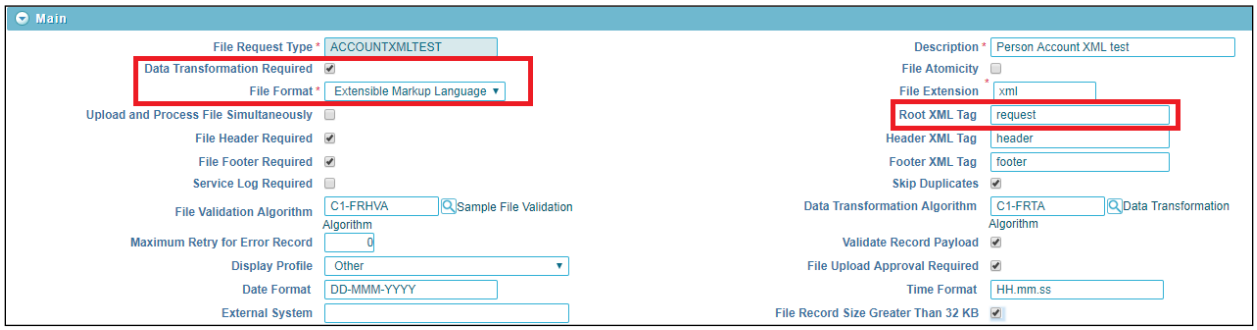


Figure 15: Root XML Tag

  - If **File Header Required** check box is selected, **Header XML Tag** is mandatory. This means service payload has header XML tag as **<request> <header>……</header>……..</request>**

Figure 16: Header XML Tag

  - If **File Footer Required** check box is selected, **Footer XML Tag** is mandatory. This means service payload has header XML tag as **<request>……..<footer>…..</footer></request>**

Figure 17: Footer XML Tag

- If **Data Transformation Required** check box is selected,
  - If **File Header Required** check box is selected, **Header Transformation** should be configured.

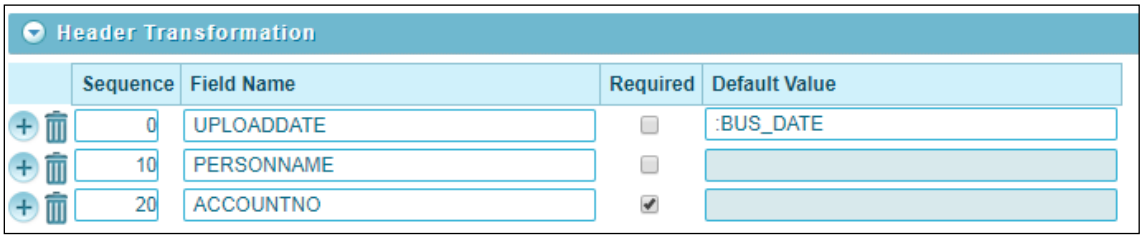


Figure 18: Data Transformation - File Header

  - If **File Footer Required** check box is selected, **Footer Transformation** should be configured.

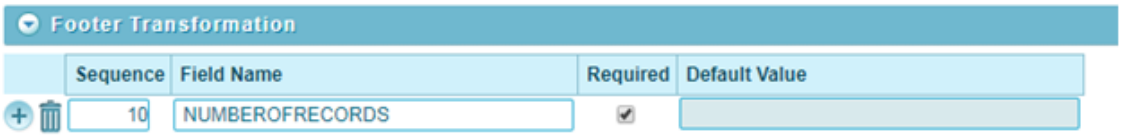


Figure 19: Data Transformation – File Footer

You can also view file header and footer details for files with **Complete** or **Pending** status using the File Upload Dashboard. For information on how to view the file header and footer details, refer to Viewing File Header and Footer Details section in File Upload Interface User Guide.

6.1.5 Ignoring or Skipping Duplicate Records in a File

To ignore or skip duplicate records in a file, select **Skip Duplicates** check box against a File Request Type. This sets the flag as True.

Figure 20: Ignore or Skip Duplicate Records

Duplicate records can be categorized into:

- Duplicate record within the same file. A sample file with duplicate records is represented below:

```
TRS|SETTLEMENT ACTIVITY|MF_FD128|2018-01-02-00.00.00||||1|0|USD|+|SUP|00081|0|0|01|N|PROCESSING|N/A|AA17L5
TRS|SETTLEMENT ACTIVITY|MF_FD128|2018-01-02-00.00.00||||1|0|USD|+|SUP|00081|0|0|01|N|PROCESSING|N/A|AA17L4
TRS|SETTLEMENT ACTIVITY|MF_FD128|2018-01-02-00.00.00||||1|0|USD|+|SUP|00081|0|0|01|N|PROCESSING|N/A|AA17L4
```

In this case, the duplicate records are uploaded with **Ignore** status (marked as **IGN**), and are not processed.

Note: This duplicity is identified on the basis of configured **File Record Identifier** in Transformation detail section of File Request Type.

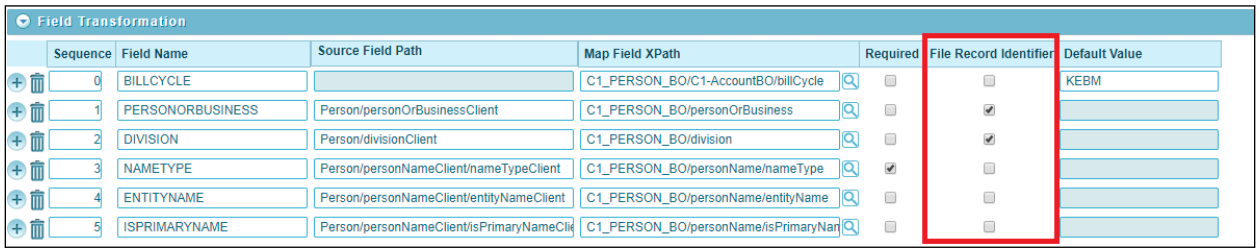


Figure 21: File Record Identifier

- Records that have been already processed in the previous file upload batch execution. There are two possible scenarios:
    - A text file has three records and is processed using file upload batch. The records are uploaded in ORMB system and the sample file appears like the image below:

```
TRS|SETTLEMENT ACTIVITY|MF_FD128|2018-01-02-00.00.00||||1|0|USD|+|SUP|00081|0|0|01|N|PROCESSING|N/A|AA17L1
TRS|SETTLEMENT ACTIVITY|MF_FD128|2018-01-02-00.00.00||||1|0|USD|+|SUP|00081|0|0|01|N|PROCESSING|N/A|AA17L2
TRS|SETTLEMENT ACTIVITY|MF_FD128|2018-01-02-00.00.00||||1|0|USD|+|SUP|00081|0|0|01|N|PROCESSING|N/A|AA17L3
```

SampleFile_1.txt

    - A text file has five records, out of which first three records have been already uploaded and processed using 'SampleFile_1.txt'. The duplicate records are uploaded with "SKIP" status and are skipped while processing. A sample file is represented below:

```
TRS|SETTLEMENT ACIVITY|MF_FD128|2018-01-02-00.00.00||||1|0|USD|+|SUP|00081|0|0|01|N|PROCESSING|N/A|AA17L1
TRS|SETTLEMENT ACIVITY|MF_FD128|2018-01-02-00.00.00||||1|0|USD|+|SUP|00081|0|0|01|N|PROCESSING|N/A|AA17L2
TRS|SETTLEMENT ACIVITY|MF_FD128|2018-01-02-00.00.00||||1|0|USD|+|SUP|00081|0|0|01|N|PROCESSING|N/A|AA17L3
TRS|SETTLEMENT ACIVITY|MF_FD128|2018-01-02-00.00.00||||1|0|USD|+|SUP|00081|0|0|01|N|PROCESSING|N/A|AA17L4
TRS|SETTLEMENT ACIVITY|MF_FD128|2018-01-02-00.00.00||||1|0|USD|+|SUP|00081|0|0|01|N|PROCESSING|N/A|AA17L5
```

SampleFile_2.txt

Note: This duplicity is identified on the basis of record existence in `CI_FILE_REQUEST_DETAIL` table using "hash key".

6.1.6 Marking the Default Status of Failed Records to "Retry" for Reprocessing

You can configure a value for **Maximum Retry for Error Record** in Main section of File Request Type window. The **Maximum Retry for Error Record** field defines the number of attempts allowed to reprocess a failed record. The value should be greater than zero.

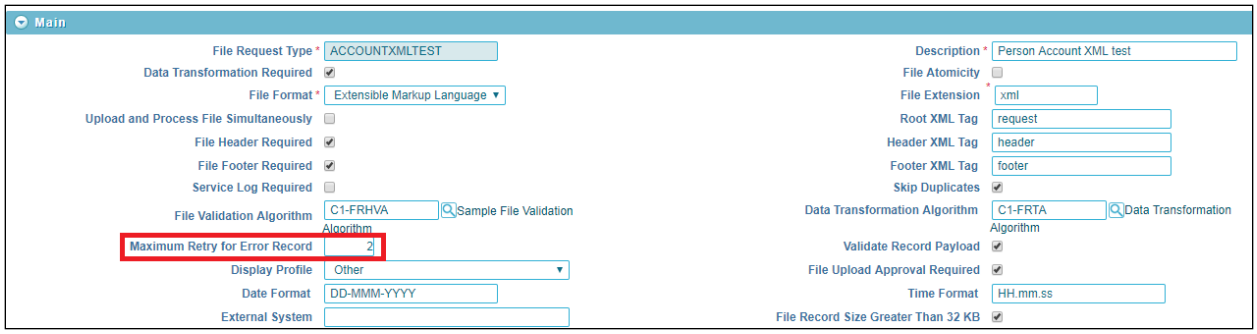


Figure 22: Update Failed Record

To configure the error messages for which the failed records should be marked with Retry status, select Retry option in **File Record Status** drop-down list in the Messages group box.

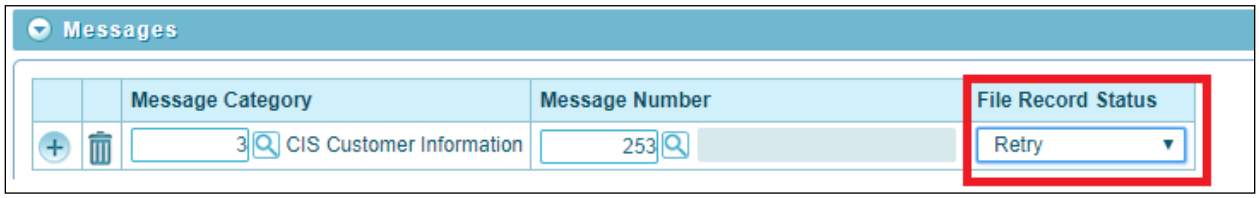


Figure 23: Update Failed Record Error Message

All failed records that are not configured in Messages section of File Request Type, are marked with **Error** status.

6.1.7 Overriding Service Level Operation

You can override service level operations using pre-processing algorithm. The pre-processing algorithm implements the **setOperation(String operation)** method to override the service level operations.

Figure 24: Override Service Level

After selecting a value in **Pre-Processing Algorithm** field, select **Add** in the Operation drop-down list to override the service level operation.

6.1.8 Skipping Service Execution

You can skip service execution for a particular record using a Pre-Processing Algorithm. The algorithm implements the **isSkipServiceExecution()** method and return “true”.

6.1.9 Executing Dependent Service

The **Dependent Service Name** field allows you to address the Payload nesting level and dependent service execution for two independent entities defined in ORMB structure. This signifies a parent-child relationship.

For example, ORMB has two independent entities, **Person** & **Account**, with each entity having their own Maintenance Object (MO). These entities further have Business Object (BO) derived from the corresponding MO. The Business Object has its own XML schema that is specific to the entity.

The legacy system allows you to generate a file with record XML having **Account** nested in **Person**. To configure the service, perform following steps:

1. Click **Add** to add a new sequence.
2. Enter sequence ID in **Sequence** field.
3. Select Business Object from the drop-down list in **Service Type** field.
4. Enter **Service Name** for which you need to configure dependent service name. For example, C1_ACCOUNT_BO
5. Specify **FK Reference** and **Pre-Processing Algorithm**.
6. Enter **Dependent Service Name**. For example, C1_PERSON_BO.

Figure 25: Dependent Service

Dependent service name configuration helps to get the parent's (Person) service Primary key as an input to its child's (Account) service pre-processing algorithm.

6.1.10 Deferring Completion of Processed Request

Use this flag to defer the completion of successfully processed request. The final status is updated once the corresponding Business Object Life cycle is completed.

It can be used for those Business Objects that have a predefined lifecycle. If a service has **Defer Completion** check box selected, then all the successfully processed records for this service are updated with 'In-Progress' status.

Note: Defer Completion is used only to update record status to legacy system.

With this status information, a legacy system can decide an action on the other dependent data upload. For example, if 'Person' Business Object has a lifecycle then, corresponding 'Account' details is not uploaded until there is an update of 'Person' Business Object lifecycle completion.

Figure 26: Defer Completion

6.1.11 Supporting Client Defined Date Format

File Upload Interface supports client-defined dates while uploading and transforming file records. You can do this by referring to the Display profile ID at the time of defining or editing or making a copy of a file request type.

Prerequisites

To link display profile ID with a file request type, you should have a Display Profile defined in the application.

Procedure

To link display profile id with file request type, you need to:

1. Enter File Request Type name.
2. Select required fields.
3. Select value from the Display Profile drop-down list.

Note: The file request should have the same date or time format as specified while defining a display profile. An individual File Request Type can support only a single date format.

Figure 27: Display Profile

6.1.12 Usage of Other Display Profile

Not all Date formats can be configured using Display Profile. To overcome this problem, use the fields 'Date Format' and 'Time Format' on File Request Type. Both these fields are visible only if you select **Other** as the Display Profile.

Figure 27.1: Display Profile selection

These field values will have the client supported Date and Time format. Both are mandatory.

Figure 27.2: Date Format and Time Format

While processing files with these date formats, they are converted to the framework supported date and time format.

6.1.13 Viewing TO DO Entries for Failed Files

If an exception occurs while executing the File Upload and Transformation batch, you can notify the user about the exception by generating a **To Do**. You can view the exception details in To Do Entry, where you see details like the total number of To Do Entries that are open, or are being worked on, or completed against each To Do Type in the file transformation and upload process.

To Do is created using its corresponding To Do Type.

Note: While defining a To Do Type, it is mandatory to set up drill key of a file ID.

Figure 28: To Do Type - Drill Keys

Once you create a To Do for capturing File Transformation and Upload Batch Error, then on submitting the respective file request, the file transformation and upload process creates a To Do for the user to review the file request. You can perform various actions or modify details of a To Do entry.

To view To Do entries:

1. From the Admin menu, select To Do and then click To Do Entry. A sub-menu appears.
2. Click Search option from the To Do Entry sub-menu. The To Do Entry Search window appears.
3. Enter a To Do Type and click Search. The grid that follows contains the To Do entries that match your criteria.

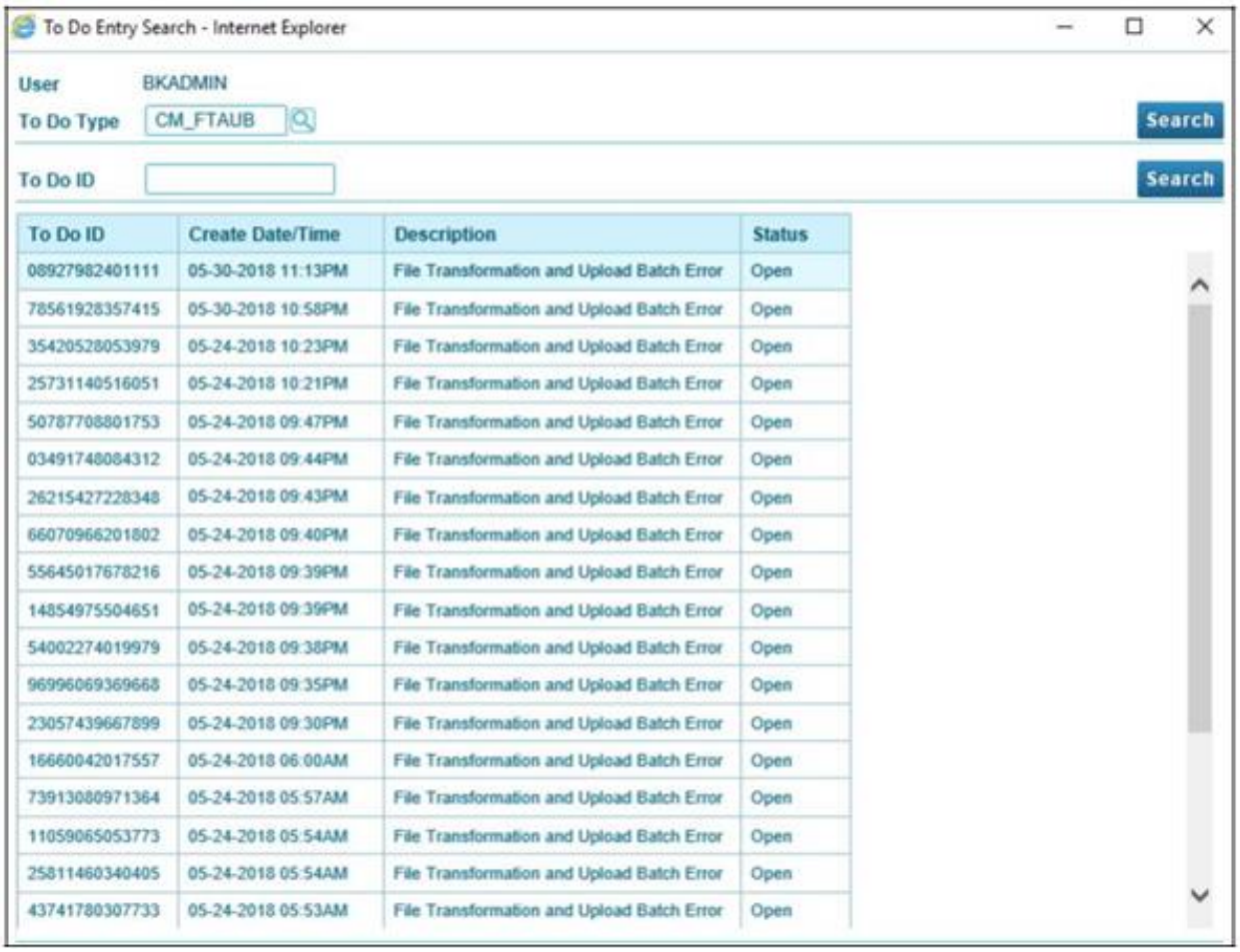


Figure 29: To Do Entry Search Results Grid

The following information appears:

Column Name	Description
To Do ID	Used to indicate the unique identifier of the To Do entry
Create Date/Time	Used to indicate the date and time the To Do entry was created by the system
Description	Used to indicate the description of the To Do Type
Status	Used to indicate the current status of the To Do entry The valid values are: • Open • Being Worked On • Complete

4. Click on text portion in any of the columns to view the respective To Do Details.

6.1.14 Post Processing Algorithm Support for File Request

You can define a post-processing algorithm, which is used to undertake some post-processing activities after successful processing of a record.

Figure 30: Post-Processing Algorithm

To attach a post-processing algorithm to file request type, specify the post-processing algorithm code in Post Processing Algorithm field within Services section.

6.1.15 Auto Generate Field Transformation

Generate button can be used to generate data transformation details for each individual service mentioned in that File Request Type.

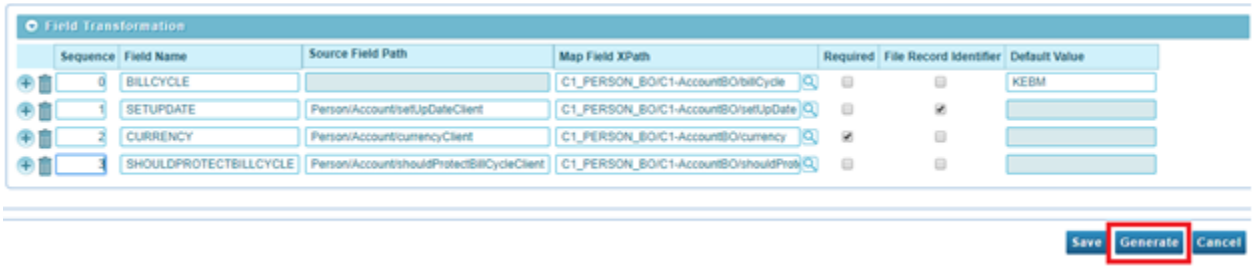


Figure 31: Auto Generate button

You can configure the number of child blocks required in the service schema. It can be zero or more.

Figure 32: Schema level configuration

Sequence Number for each field will be based on the defined order in schema. For fixed position file format, Field, start position and end position will be populated using its defined order in schema and its defined precision i.e. length.

You can edit auto generated data transformation details. For any addition or updation or deletion of field details, data transformation Sequence will be auto reset.

If we don't want to go with sequence auto reset, then, uncheck the "Transformation Auto Sequence Reset" checkbox under File Request Type. This field on File Request Type will be only visible after auto generation of data transform details.

6.2 Approval Workflow configuration for File Upload Interface

Steps for configuration of approval workflow for File Upload Interface

6.2.1 Create Approval Workflow Group

1. Go to Admin Menu > A > Approval Workflow Group.
2. Click on Add button.

3. Enter the details and click on save button.

   Main group box:

Field Name	Description
Approval Transaction Type	Business Object Based
Approval Workflow Group	FILERQUPLD
Display UI Map	C1-FileUploadApprovalDisplayMp
Input UI Map	C1-FileUploadApprovalDisplayMp
Approval Algorithm	C1-APPALG
Approval Post Processing Algorithm	C1-FILERQAPP
Transaction Creation Algorithm	C1-AXCREATE
Data Retrieval Algorithm	C1_APPDATA

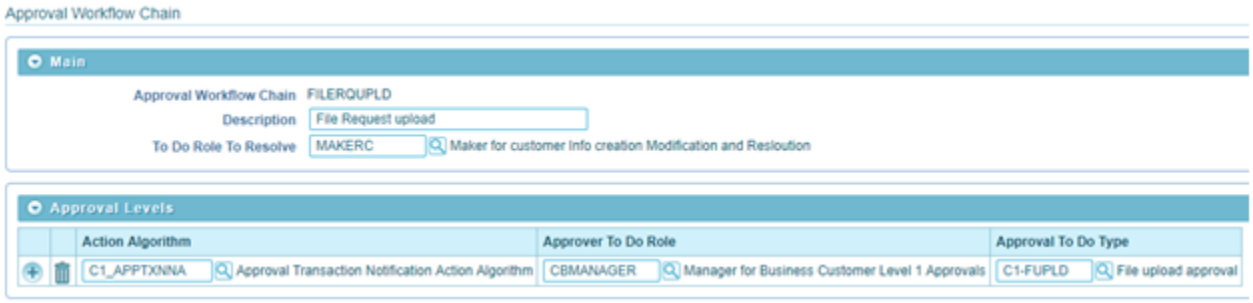
Business Objects and Group BO Relation:

Field Name	Description

Business Object	C1-FileRq
List	FALSE

6.2.2 Create Approval Workflow Chain

1. Go to Admin Menu > A > Approval Workflow Chain and click Add.



2. Enter the required details and click Save.

Main group box:

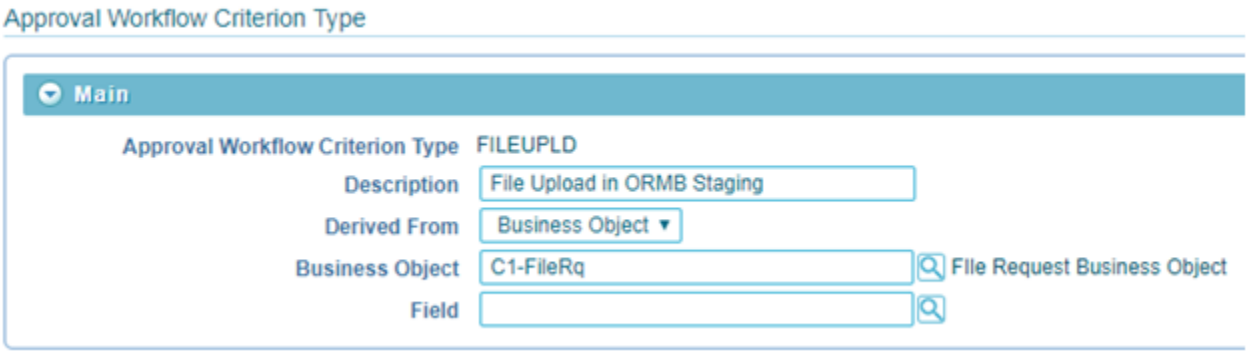
Field Name	Description
Approval Workflow Chain	FILERQUPLD
To Do Role To Resolve	MAKERC

Approval Levels group box:

Field Name	Description
Action Algorithm	C1_APPTXNNA
Approval To Do Type	C1-FUPLD

6.2.3 Create Approval Workflow Criterion Type

1. Go to Admin Menu > A > Approval Workflow Criterion Type and click Add.



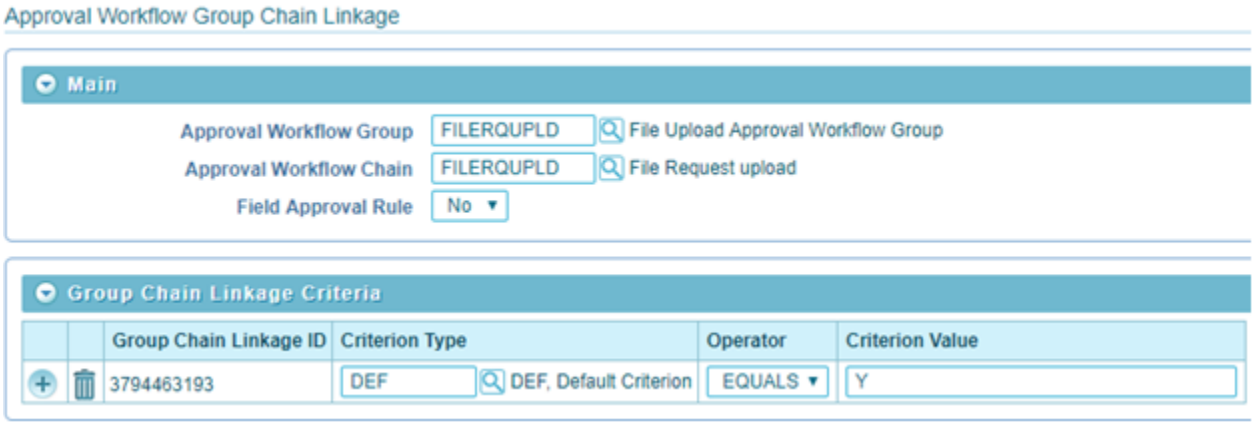
2. Enter the required details and click on Save.

Main group box:

Field Name	Description
Approval Workflow Criterion Type	FILEUPLD
Derived From	Business Object
Business Object	C1-FileRq

6.2.4 Create Approval Workflow Group Chain Linkage

1. Go to Admin Menu > A > Approval Workflow Group Chain Linkage and click Add.
2. Enter the required details and click on Save.



Main group box:

Field Name	Description
Approval Workflow Group	FILERQUPLD
Approval Workflow Chain	FILERQUPLD
Field Approval Rule	No

Group Chain Linkage Criteria:

Field Name	Description
Criterion Type	DEF
Operator	EQUALS
Criterion Value	Y

6.2.5 Create Approval Workflow Settings

1. Go to Admin Menu > A > Approval Workflow Settings and click Add.
2. Enter the required details and click Save.

Main group box:

Field Name	Description
Approval Workflow Group	FILERQUPLD
Approval Chain Selection Algorithm	C1-APPCHAIN
Display UI Map	No
Input UI Map	Yes
Approval Algorithm	Yes

6.2.6 Update “File Upload Approval Required” flag as true

7. Updating Records marked with 'Error' or 'Pending' status

When updating records marked with 'Error' or 'Pending' status, you have two possible modes:

- Update records marked with 'Error' status to 'Retry' status
- Update records marked with 'Pending' status to 'Error' status

ORMB provides you two options to update records.

- File Upload Dashboard
- File Request Status Update Batch

7.1 File Upload Dashboard

This is used to update status for less number of records. To update records with "Error" to "Retry" status, you need to follow below steps:

1. Click the Menu link in the Application toolbar. A list appears.
2. Select Tools from the list. A sub-menu appears.
3. Click File Upload Dashboard option. The Search File window appears.
4. Enter either File Name or select File Request Type from the drop-down list. Click Search.
5. The file details that meet the search criteria appear in the Search Results section in a tabular format.

Figure 33: File Upload Dashboard

6. The counts in respective Status columns represent the number of records. The numbers are linked to Search File Record Detail form.
7. Click the number in Error column to view the record details.

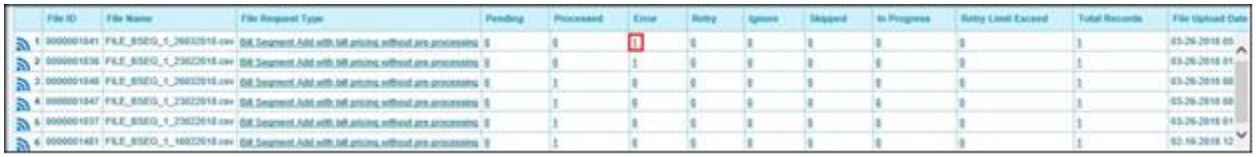


Figure 34: Error Records

8. File Records window appears. Select records for which "Retry" status is to be updated.

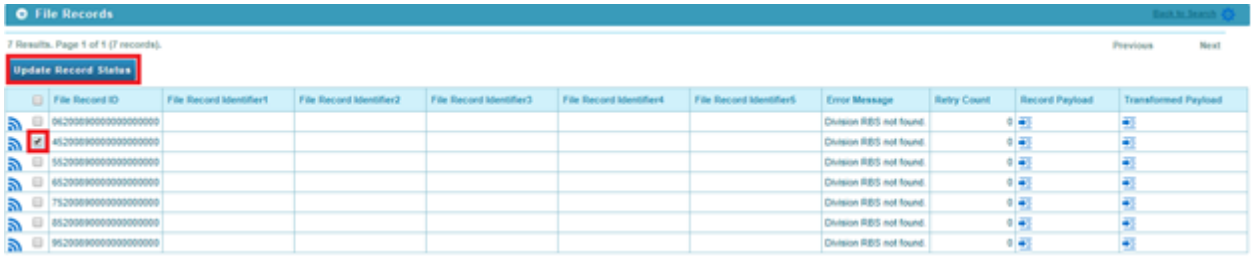


Figure 35: File Records

9. Click Update Record Status. The File Request Detail Update Reason dialog box appears.

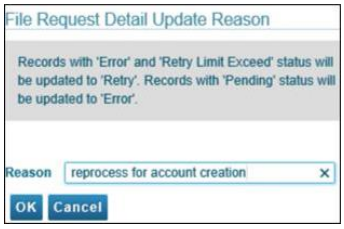


Figure 36: File Request Detail Update Reason

10. Specify the reason and click OK. Similarly, you need to follow above mentioned steps to update records with “Pending” to “Error” status.

Note: These is an online process for bulk update, but it could have a performance impact.

7.2 File Request Status Update Batch

This batch can be used for bulk record status update from “Error” to “Retry” status or “Pending” to “Error” status.

8. Transforming Data

8.1 Transforming File Record in Client Supported Format into ORMB Conform XML

You can transform files in Comma Separated Values (CSV) or Extensible Markup Language (XML) or Fixed Position or JavaScript Object Notation (JSON) or Pipe Separated Values (PSV) or Tilde Separated Values (TSV) formats to ORMB conforming XML using “Data Transformation Algorithm”.

“Data Transformation Algorithm” implements “FileRequestTranformationAlgorithmSpot”.

To transform file record, specify File Request Transformation algorithm in Data Transformation Algorithm field present in Main section. You can also use the Search ( ) icon corresponding to Data Transformation Algorithm field.

Note: Ensure that Data Transformation Required field is selected before specifying the algorithm.

Figure 37: Data Transformation Algorithm

Product provides default transformation algorithm capabilities of the same are listed below.

8.2 Configuring a Default Value in Data Transformation

You can configure default value corresponding to a field. The default value for Sequence should be zero.

Note: Sequence zero is reserved for default value configuration.

To set default value for fields with ‘Date’ as datatype, you can either provide a date value in following date format ‘yyyy-MM-dd / yyyy-MM-dd-HH.mm.ss’or the predefined values listed below.

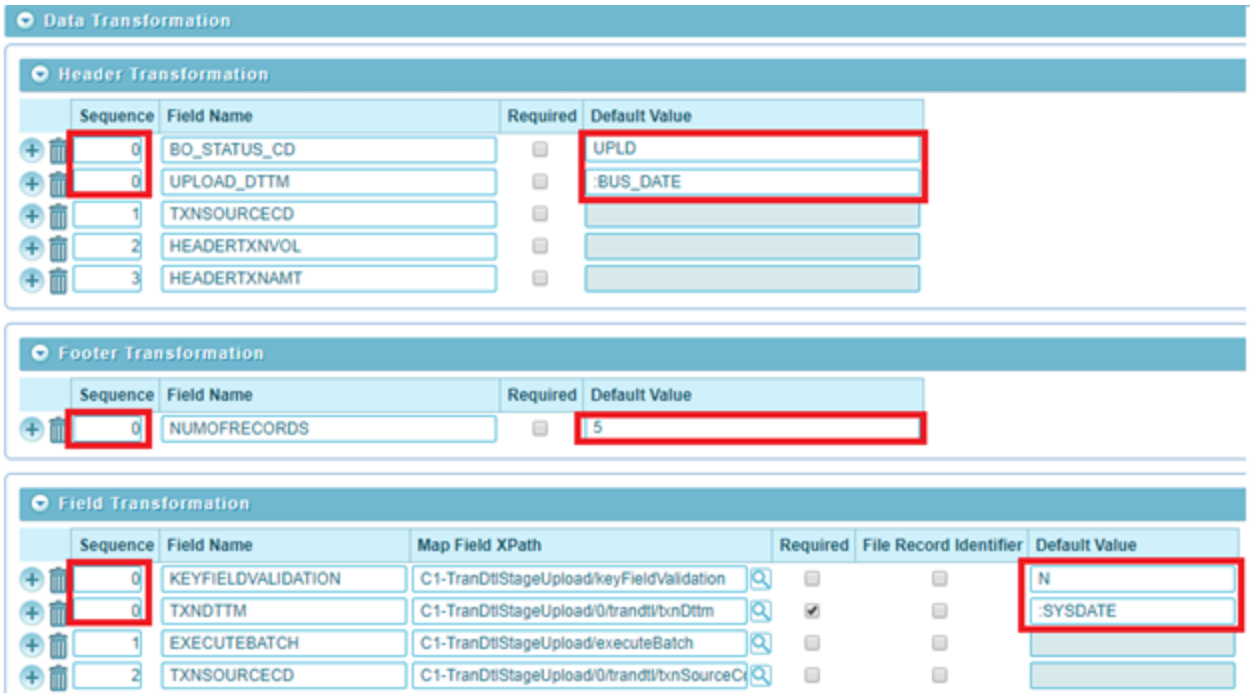


Figure 38: Data Transformation - Default Value

- :BUS_DATE for Business Date - relates to process date
- :SYSDATE for System date - relates to System Date
- :STD_DATE for Standard date - relates to LOCALE date
- :BUS_DTTM for Business Date time - relates to process date time
- :SYS_DTTM for System date time - relates to System Date Time
- :STD_DTTM for Standard date time - relates to LOCALE date time

To update File Request Type detail, click Edit ( ) icon. This enables the 'Default Value' field and fetches the datatype of that corresponding field. The updated value is then used to set the default value.

8.2.1 Applying Default Values Set in File Validation Algorithm to a Field

File Upload Interface provides you with set of five parameters which you can use while transforming records. These parameters are defined using setter methods for default1/2/3/4/5 in file validation algorithm.

Prerequisite

To refer default parameter values, you should have a File Validation Algorithm attached in Main section.

Procedure

To refer these default parameters, you can use either of the following constants in Default Value field within Transformation Details section.

- :DEFAULT1/2/3/4/5
- :SYSDATE/SYS_DTTM
- :BUS_DATE/BUS_DTTM
- :STD_DATE/STD_DTTM

When you use any of the above constants, the value corresponding to the defined parameter is set as default value to the respective field name.

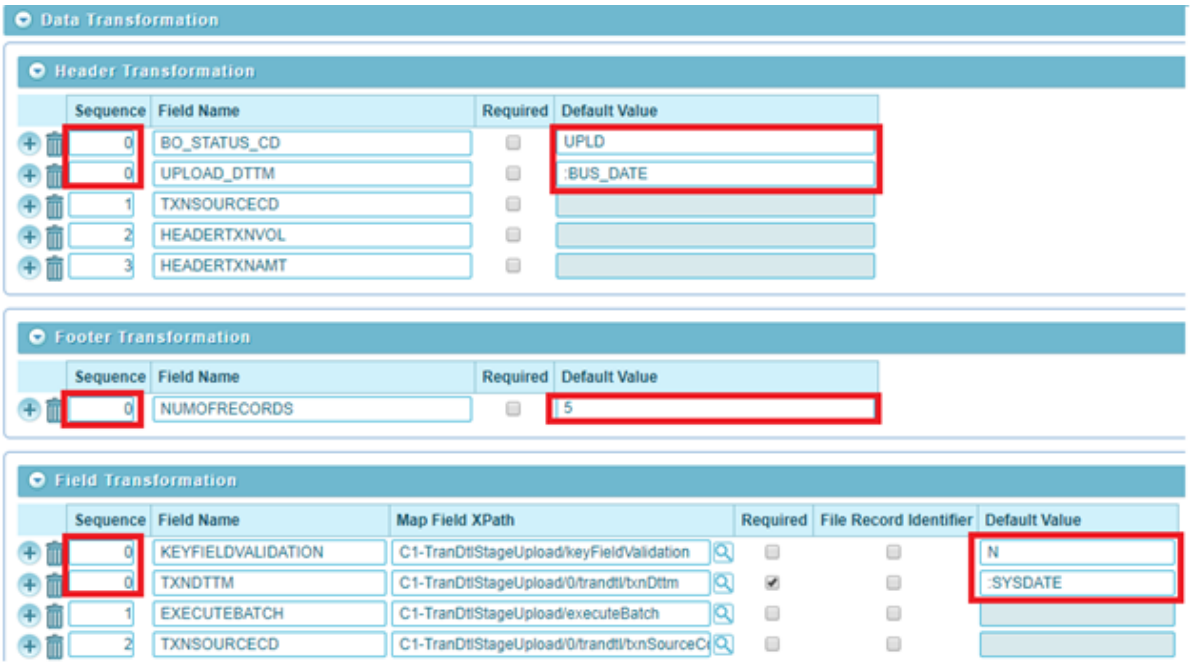
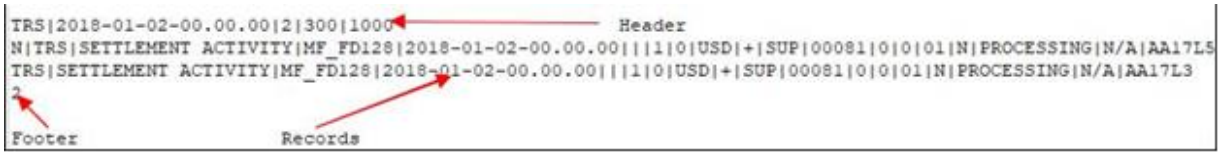


Figure 39: Applying Default Values Set in File Validation Algorithm to a Field

8.3 Mapping Data Line or Record in a File

You can map record in a file using Data Transformation section. Using Data Transformation configuration, mapping can be done only for files in CSV or XML or Fixed Position or JavaScript Object Notation (JSON) or Pipe Separated Values (PSV) or Tilde Separated Values (TSV) formats.

A sample file in PSV format is represented below. The entries are separated by '|' (pipe).



Sample_1.dat

File is divided into three segments:

- File Header - First row in the file
- File record(s)
- File Footer - Last row in the file

The File Request Type-Data Transformation is divided in 3 sections:

- Header Transformation
- Footer Transformation
- Field Transformation

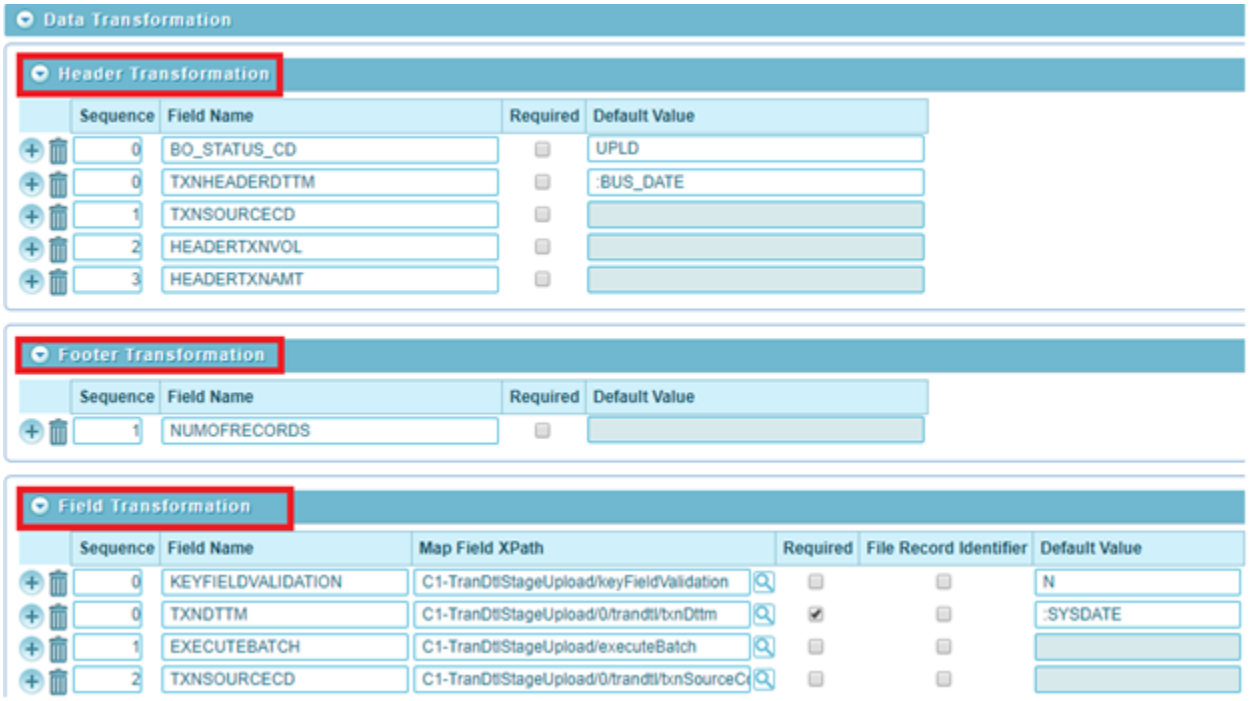


Figure 40: Mapping data line

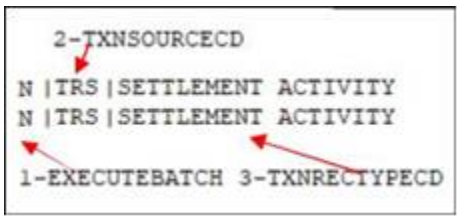
All three segments in data transformation configuration are independent of each other. You can configure default values for any of these segments.

Multiple number of file records should have the same number of fields. For example, if a file has 10 records with seven fields, then all those 10 records should have seven fields, with or without values.

There are two different ways to configure data line mapping:

- For files in CSV or PSV or TSV format:
    - Mapping is done by defining **Sequence**

A sample file is represented below:



Sample File.txt

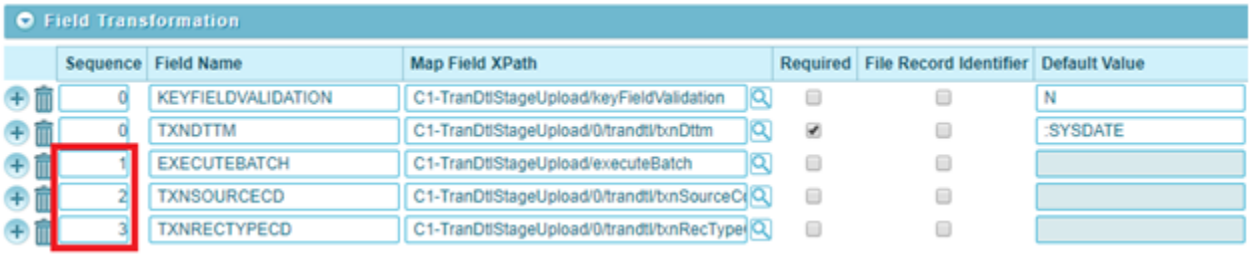


Figure 41: Sample File Mapping

- For file data mapping, Sequence index starts with 1
- Sequence with Zero index is reserved for mapping default value

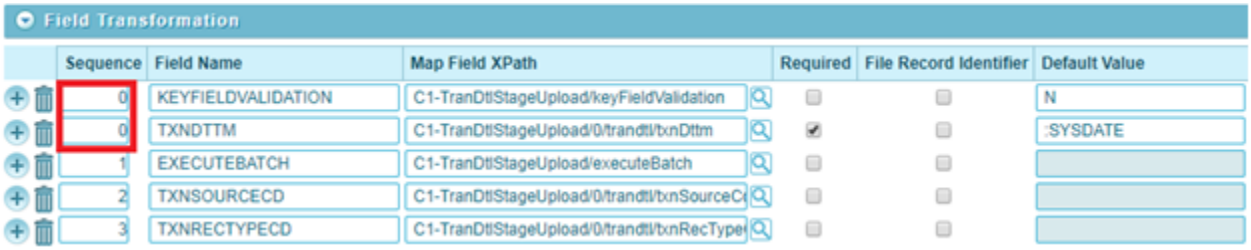


Figure 42: Sample File Mapping Default Value

- Same Sequence mapped for multiple fields

- For files in 'Fixed Position' format:
  - Mapping is done using 'Start Position' and 'End Position'.

Figure 43: File Mapping – Fixed Position

- Start Position index starts with 1
- Sequence with **Zero** index used for mapping default value

> Note that it is not required to define 'Start Position' and 'End Position'.

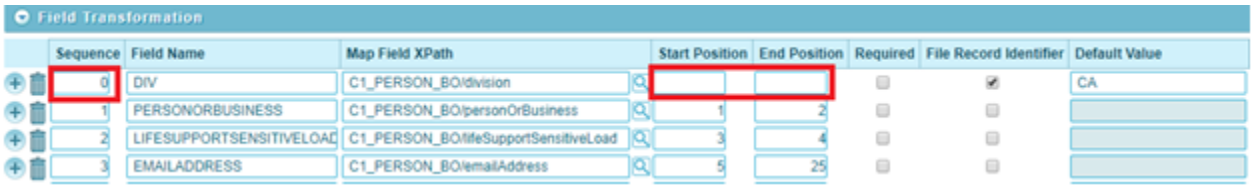


Figure 44: File Mapping – Sequence Index

- Same positions can be mapped for multiple fields

8.3.1 Mapping XML/JSON Data Line or Record in a File

You can map record in a XML file using Data Transformation section. A sample file in XML format is represented below.

```
<root>
    <header>
        <personName>MaheshK</personName>
        <accountNo>4321</accountNo>
    </header>
    <footer>
        <numberOfRecords>4</numberOfRecords>
    </footer>
    <request>
        <Person>
            <personOrBusinessClient>R</personOrBusinessClient>
            <divisionClient>CA</divisionClient>
            <personNameClient>
                <nameTypeClient>PRIM</nameTypeClient>
            </personNameClient>
        </Person>
    </request>
</root>
```

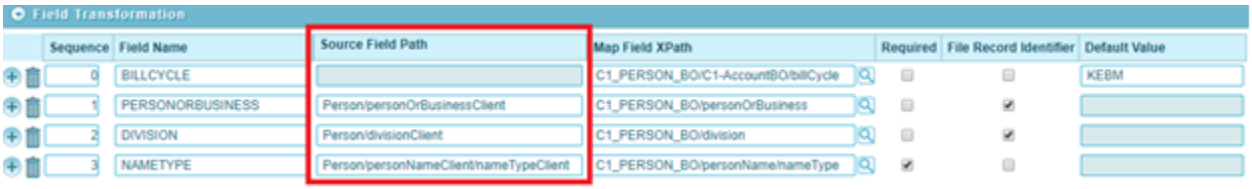
Sample_1.dat

Every record in XML file is divided into three blocks:

- **Header**: Defined in a block having tag name that is defined as **Header XML Tag** in its corresponding File Request Type
- **Service Data**: Defined in a block having tag with service/schema name
- **Footer**: Defined in a block having tag name that is defined as **Footer XML Tag** in its corresponding File Request Type

For XML/JSON record mapping, Source Field Path in Field Transformation is used to hold field xpath in legacy XML/JSON file.

- Source Field Path in Field Transformation is visible only if file format is either **XML** or **JSON**



8.4 Mapping File Sequence

This is a unique user defined field, which holds the mapped file sequence field value. It is used as column identifier of a file record. It is free text and can have any user defined value. It has to be uniquely defined within its respective sections i.e. Header Transformation or Footer Transformation or Field Transformation.

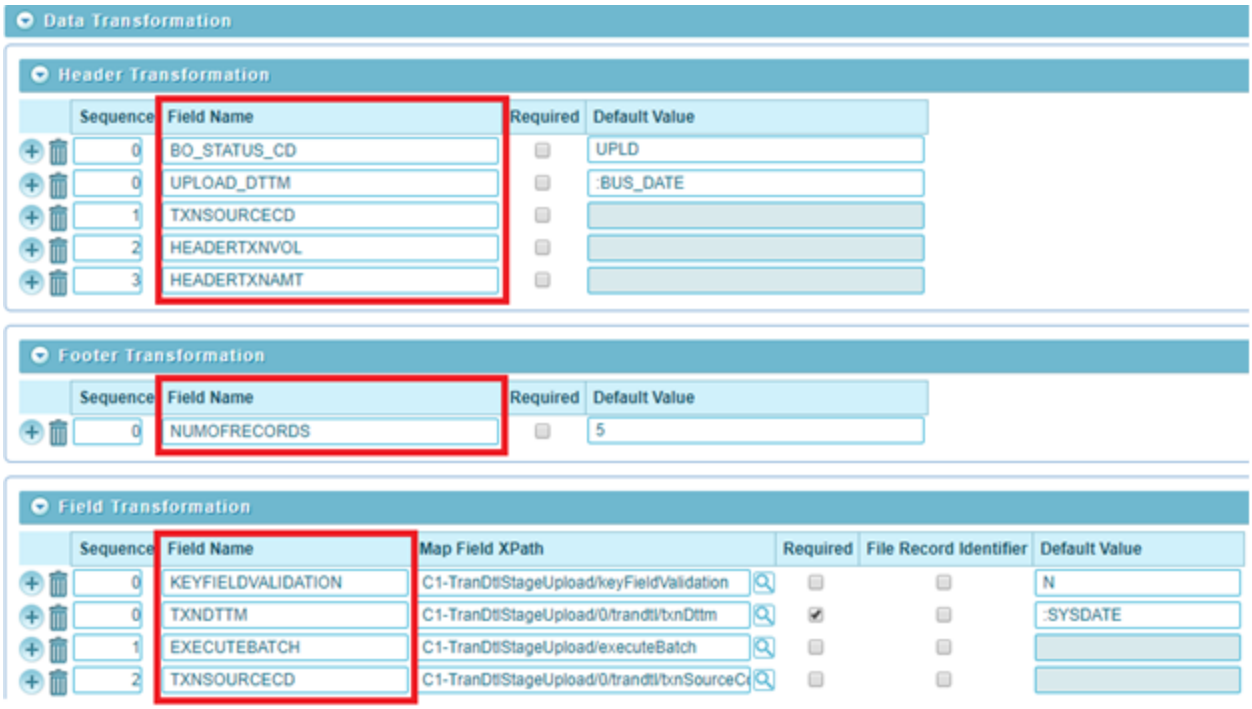


Figure 45: Data Transformation – Field Name

8.5 Using Map Field XPath to Transform Records

The Map Field XPath uses ORMB application provided sample Data Transformation Algorithm to transform file record into ORMB conform XML. If you want to use this algorithm, you must provide “Map Field XPath” for every configured field.

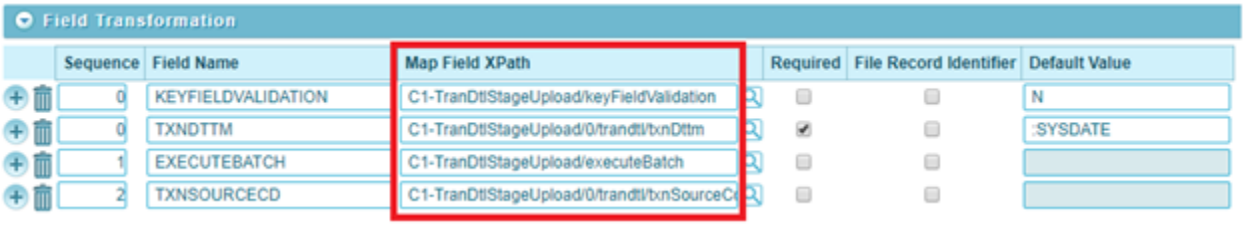


Figure 46: Data Transformation - Map Field Xpath

Note: The Header Transformation and Footer Transformation does not have ‘Map Field XPath’ column.

These will have an ‘XPath’ value with reference to its configured service schema.

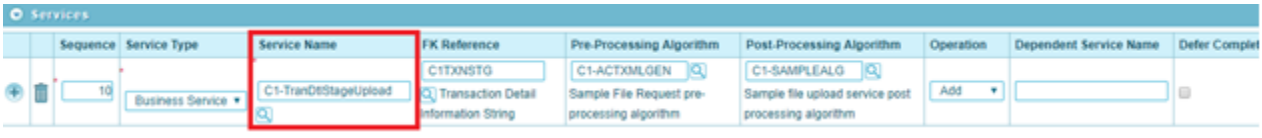


Figure 47: Service Name – Xpath Value

Note: You can search ‘xpath’ values using the Search ( ) icon or enter a free text.

Index should be used to configure child entity element xpath in a service schema. To configure child entity element xpath, you need to follow below steps:

1. Click Search corresponding to Map Field XPath. File request transform map field zone appears.
2. Select Service Type from the drop-down list.
3. Enter Service Name.

Note: You can select only the child element xpath with index value zero. If you require more than one child elements for a single entity, then you need to select child element xpath with zero' index and edit the required index. For example, '1','2', etc.

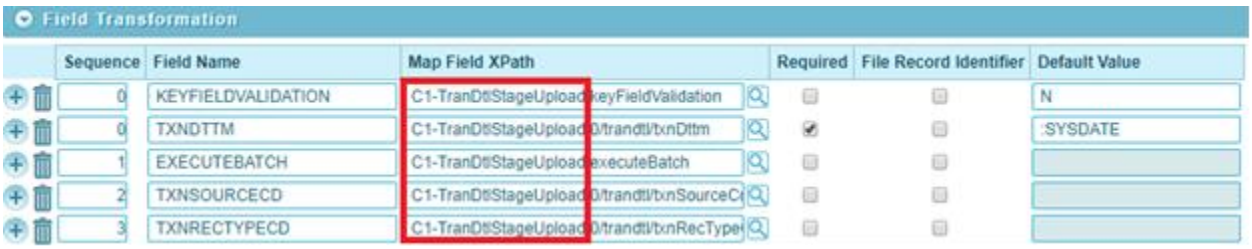


Figure 48: Map Field Xpath

8.6 Link Parent Service Output with Child Service Input

File Upload Interface provides you the flexibility to link parent service output (Primary) key values to child service input field values while transforming data. The supported mapping format is **':PK1/2/3/4/5[CHILD_SERVICE_NAME=PARENT_SERVICE_NAME]'**

For example, there are three business objects which form a hierarchy:

- Person
- Account
- Service Agreement

To map 'Per_Id' within 'Account' business object with primary key value of 'Person' business object (Per_Id), enter the value **:PK1[C1-AccountBO=C1_PERSON_BO]** in **Default Value** field corresponding to Person ID field.

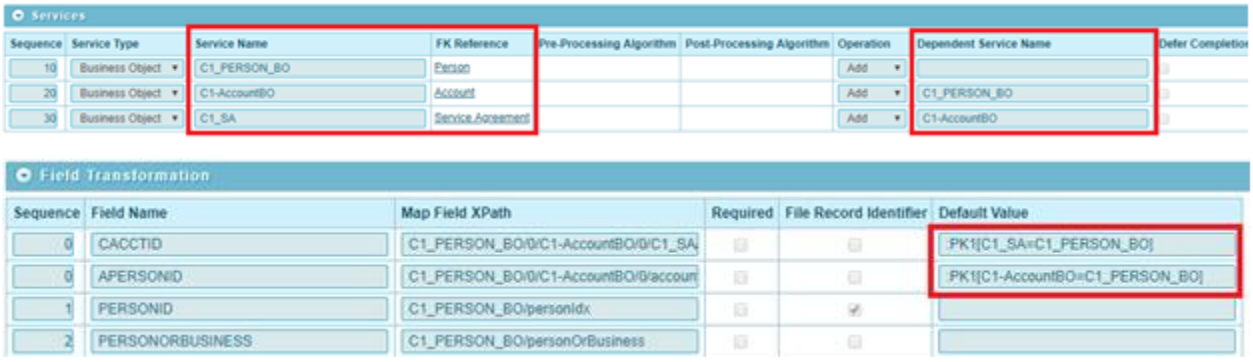


Figure 49: Mapping Primary Key Value of 'Person' within 'Account' Business Object

To map 'C1_SA' within 'Service Agreement' business object with primary key value of 'Account' business object, enter the value **:PK1[C1_SA=C1-AccountBO]** in Default Value field corresponding to Account ID field.

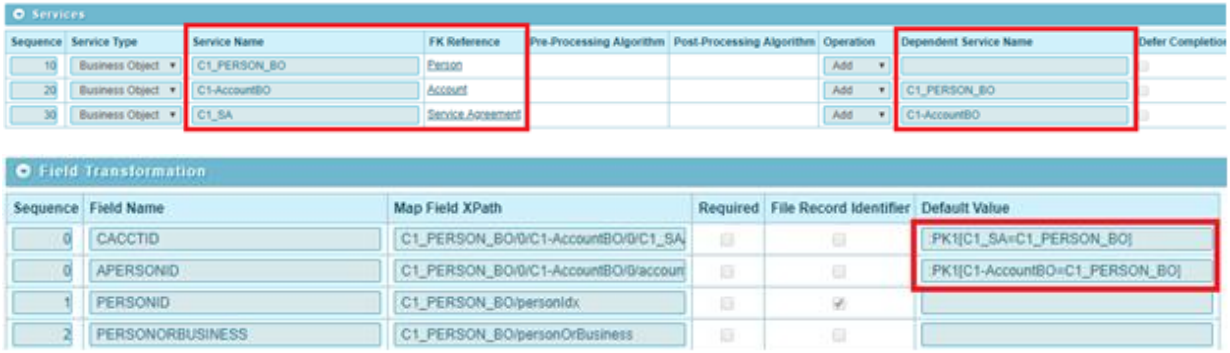


Figure 50: Mapping Primary Key Value of 'Service Agreement' within 'Account' Business Object

Linking parent service output with child service input helps to avoid use of pre-processing algorithm.

8.7 Validating Input Values

You can perform mandatory field level validations for every record while uploading data using File Transform and Upload (C1-FTRAN) Batch.

To perform field level mandatory validations, select Required check box for those required fields in its corresponding section.

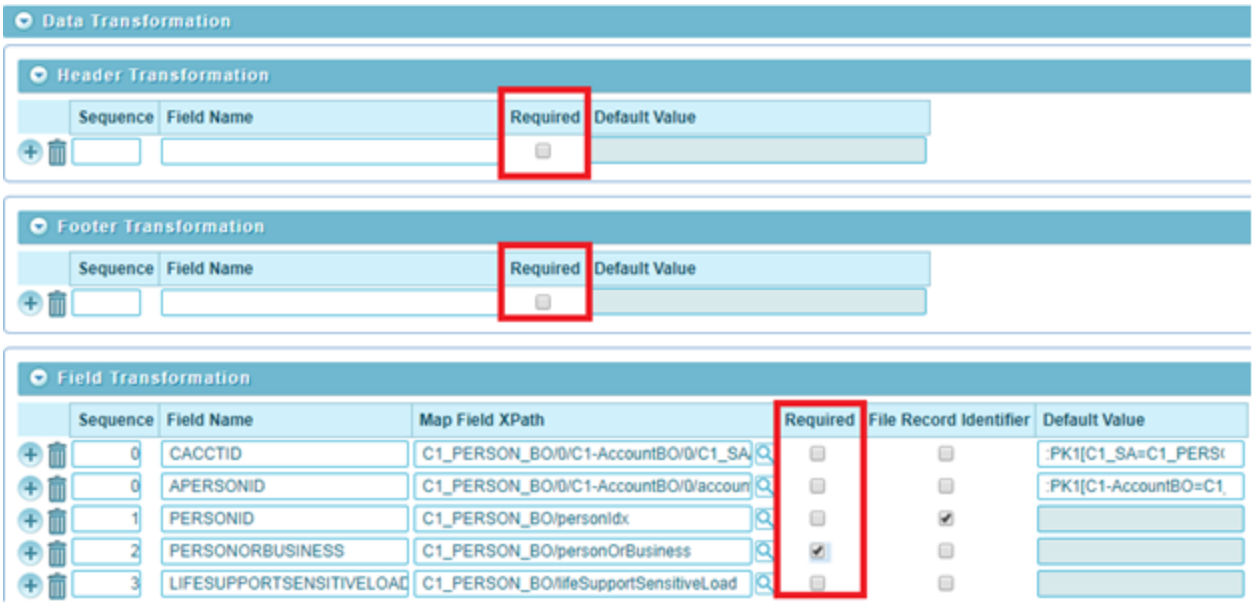


Figure 51: Validating Input Values

9. File Management System

9.1 Overview of files uploaded on SFTP server

You can have an overview of list of files uploaded on SFTP server. In addition to these, this list will also have those files in ORMB staging with 'Pending' and 'Approval Pending' status i.e. files that has been uploaded in ORMB staging but not processed.

File overview is categorized into four different file status,

- **Ready To Upload**: These files available to upload in ORMB staging
- **Copy in Progress**: File upload on SFTP server is in progress
- **Approval Pending**: File uploaded in ORMB staging and awaiting for approval required to process the file
- **Pending to Process**: File uploaded in ORMB staging and now available to process

List of files to be displayed with following details,

- **File Name** – Name of the file
- **File Request Type** – This column have value for only those files that has already been uploaded in ORMB staging with 'Pending' and 'Approval Pending' status
- **File Path** – File Path for files that have already been uploaded in ORMB staging is always 'ORMB Staging'
- **File Status** - Refer the list above
- **File Size** – File Size is shown for only those files that has not yet uploaded in ORMB staging
- **File Upload Approval Required**: Flag to show whether approval is required to process this file
- **Approval Transaction ID**: Approval transaction ID for the file (You can navigate to the corresponding approval transaction)
- **Upload Date Time**: File upload date time on SFTP server

Figure 52: File Management System

10. Rule configuration for SFTP poller batch

SFTP poller batch has the flexibility to poll the files on SFTP server that are to be uploaded in ORMB staging using rule configuration. There are two predefined rule types used in SFTP poller batch execution:

- FILE_RQ_TYPE - File Upload Rule Type
- FILE_UP_FAIL - File Upload Rule Failure Type

Figure 53: Rule Type

10.1 Rule Type for polling files on SFTP server

Prerequisites

- DBMS Scheduler Job should be defined for post execution of the required batch program
- Email configuration using Admin Menu > M > Master Configuration > Email Sender Configuration

File Upload Rule Type (FILE_RQ_TYPE) has the rules defined for polling those files that should be uploaded in ORMB staging. Available rule Input and Output parameters for rule type File Upload Rule Type (FILE_RQ_TYPE) are:

Rule Parameter	Type	Description
File Name (FILE_NAME)	Input	This field will have the name of the picked file.
File Path (FILE_PATH)	Input	This field will have the directory path of the picked file.
File Extension (FILE_EXTENSION)	Input	This field will have the extension of the picked file i.e. txt, csv, xml etc.
Directory Name (DIRECTORY_NAME)	Input	This field will have the directory name of the picked file.
Number Of Files (NUMBER_OF_FILES)	Input	This field will have the count for number of files available in the directory of picked file.
Directory Contains File (DIRECTORY_CONTAINS_FILE)	Input	This field will have the comma separated list of file names that are available in the same directory that of the picked file.
No Of Supported Extension Files (NO_OF_SUPPORT_EXTN_FILES)	Input	This field will have the count for number of only those files having the extension that is defined in those configured File Request Type's.

Rule Parameter	Type	Description
No Of Upload Confirmation Files (NO_OF_UPLD_CONFIRM_FILES)	Input	This field will have the count for number of only those files having the extension that is not defined in those configured File Request Type's.
Alert Notification Time (ALERT_NOTIFY_TIME)	Input	This field will have the current execution time (Format-HH:mm).
Job Name (JOB_NAME)	Output	This field should have the Job name that has the batch program that is required to be executed.
Dependent Job Name (DEPENDENT_JOB_NAME)	Output	This field should have the dependent Job Name i.e. the job name that has been already done which will allow to execute the current job defined against Job Name (JOB_NAME)
Dependent Job Interval In Minutes (DEPENDENT_JOB_INTERVAL_MIN)	Output	This field should have the execution time interval in minutes of dependent Job i.e. Dependent job done within this period of time.
Message Category (MESSAGE_CAT_NBR)	Output	This field should have message category number required for creation of email body content.
Message Number (MESSAGE_NBR)	Output	This field should have message number that has the email body content.
Parameter 1 (MESSAGE_PARM1)	Output	This field should have the message parameter value 1.
Parameter 2 (MESSAGE_PARM2)	Output	This field should have the message parameter value 2.
Parameter 3 (MESSAGE_PARM3)	Output	This field should have the message parameter value 3.
Parameter 4 (MESSAGE_PARM4)	Output	This field should have the message parameter value 4.
Parameter 5 (MESSAGE_PARM5)	Output	This field should have the message parameter value 5.
Notify Email (NOTIFY_EMAIL)	Output	This field should have value either 'true' or 'false' which will decide to send an email notification
Email To Recipients (EMAIL_TO_RECIPIENTS)	Output	This field should have comma separated list of email 'To' recipients.

Email Cc Recipients (EMAIL_CC_RECIPIENTS)	Output	This field should have comma separated list of email 'CC' recipients.
Subject (EMAIL_SUBJECT)	Output	This field should have the subject line of the email to be send.
Email Body Content (EMAIL_BODY_CONTENT)	Output	This field should have the email body content of the email to be send.
Last Notified Time (LAST_NOTIFIED_TIME)	Output	This field value will allow us to avoid resending of multiple mails or creation of TODO on succession of the same rule. This field should have the time (Format-HH:mm) that is in sync with that of alert notification time provided in rule condition i.e. Alert Notification Time (ALERT_NOTIFY_TIME) after a reach (>=) of this time mail will be triggered.
To Do Type (TODO_TYPE)	Output	This field should have the To Do Type for which a To Do will be created.
Role ID (TODO_ROLE)	Output	This field should have the To Do Role used for creating To Do.
Drill Key 1 (DRILL_KEY1)	Output	This field should have the drill key 1 corresponding to the given To Do Type which will be used for creating To Do.
Drill Key 2 (DRILL_KEY2)	Output	This field should have the drill key 2 corresponding to the given To Do Type which will be used for creating To Do.
Drill Key 3 (DRILL_KEY3)	Output	This field should have the drill key 3 corresponding to the given To Do Type which will be used for creating To Do.
Sort Key (SORT_KEY)	Output	This field should have the sort key corresponding to the given To Do Type which will be used for creating To Do.
Assign to User (ASSIGNED_TO)	Output	This field should have the User Id to which a created To Do will be assigned

Note: All those Rule output parameters corresponding to Message Category and Number are optional. You can use them as per the requirement to form an email body content.

BATCH_NUM: This is the constant having current batch number as a value. This constant can be used as a Rule output parameter value for any of the 'DRILL_KEY1/2/3' or 'EMAIL_SUBJECT' or 'EMAIL_BODY_CONTENT'.

FILE_NAME: This is the constant having input file name as a value. This constant can be used as a Rule output parameter value for any of the 'MESSAGE_PARM1/2/3/4/5' or 'EMAIL_SUBJECT' or 'EMAIL_BODY_CONTENT'.

10.2 Rule Type for failure notification

Prerequisite

- Email configuration using Admin Menu>>M>>Master Configuration>>Email Sender Configuration

File Upload Rule Failure Type (FILE_UP_FAIL) rule type has the rules defined for sending email notification for the failure in file polling using File Upload Rule Type (FILE_RQ_TYPE). Available rule Input and Output parameters for rule type File Upload Rule Failure Type (FILE_UP_FAIL) are:

Rule Parameter	Type	Description
File Name (FILE_NAME)	Input	This field will have the name of the picked file.
File Path (FILE_PATH)	Input	This field will have the directory path of the picked file.
File Extension (FILE_EXTENSION)	Input	This field will have the extension of the picked file i.e. txt, csv, xml etc.
Directory Name (DIRECTORY_NAME)	Input	This field will have the directory name of the picked file.
Number Of Files (NUMBER_OF_FILES)	Input	This field will have the count for number of files available in the directory of picked file.
Directory Contains File (DIRECTORY_CONTAINS_FILE)	Input	This field will have the comma separated list of file names that are available in the same directory that of the picked file.
No Of Supported Extension Files (NO_OF_SUPPORT_EXTN_FILES)	Input	This field will have the count for number of only those files having the extension that is defined in those configured File Request Type's.
No Of Upload Confirmation Files (NO_OF_UPLD_CONFIRM_FILES)	Input	This field will have the count for number of only those files having the extension that is not defined in those configured File Request Types.
Alert Notification Time (ALERT_NOTIFY_TIME)	Input	This field will have the current execution time (Format-HH:mm).

Message Category (MESSAGE_CAT_NBR)	Output	This field should have message category number required for creation of email body content.
Message Number (MESSAGE_NBR)	Output	This field should have message number that has the email body content.

Rule Parameter	Type	Description
Parameter 1 (MESSAGE_PARM1)	Output	This field should have the message parameter value 1.
Parameter 2 (MESSAGE_PARM2)	Output	This field should have the message parameter value 2.
Parameter 3 (MESSAGE_PARM3)	Output	This field should have the message parameter value 3.
Parameter 4 (MESSAGE_PARM4)	Output	This field should have the message parameter value 4.
Parameter 5 (MESSAGE_PARM5)	Output	This field should have the message parameter value 5.
Notify Email (NOTIFY_EMAIL)	Output	This field should have value either 'true' or 'false' which will decide to send an email notification
Email To Recipients (EMAIL_TO_RECIPIENTS)	Output	This field should have comma separated list of email 'To' recipients.
Email Cc Recipients (EMAIL_CC_RECIPIENTS)	Output	This field should have comma separated list of email 'CC' recipients.
Subject (EMAIL_SUBJECT)	Output	This field should have the subject line of the email to be send.
Email Body Content (EMAIL_BODY_CONTENT)	Output	This field should have the email body content of the email to be send.
Last Notified Time (LAST_NOTIFIED_TIME)	Output	This field value will allow us to avoid resending of multiple mails or creation of TODO on succession of the same rule. This field should have the time (Format-HH:mm) that is in sync with that of alert notification time provided in rule condition i.e. Alert Notification Time (ALERT_NOTIFY_TIME) after a reach (>=) of this time mail will be triggered.
Missing File Name (MISSING_FILE_NAME)	Output	This field will have the pipe separated list of expected file names within the current directory of picked file.
Missing File Description (MISSING_FILE_DESC)	Output	This field will have the pipe separated list of file description. This list should be corresponding to that of 'Missing File Name' list. This file description will be used to populate **[%FILE_NAMES]** constant.

Rule Parameter	Type	Description
To Do Type (TODO_TYPE)	Output	This field should have the To Do Type for which a To Do will be created.
Role ID (TODO_ROLE)	Output	This field should have the To Do Role used for creating To Do.
Drill Key 1 (DRILL_KEY1)	Output	This field should have the drill key 1 corresponding to the given To Do Type which will be used for creating To Do.
Drill Key 2 (DRILL_KEY2)	Output	This field should have the drill key 2 corresponding to the given To Do Type which will be used for creating To Do.
Drill Key 3 (DRILL_KEY3)	Output	This field should have the drill key 3 corresponding to the given To Do Type which will be used for creating To Do.
Sort Key (SORT_KEY)	Output	This field should have the sort key corresponding to the given To Do Type which will be used for creating To Do.
Assign to User (ASSIGNED_TO)	Output	This field should have the User Id to which a created To Do will be assigned

Note: All those Rule output parameters corresponding to Message Category and Number are optional. These can be used as per the requirement to form an email body content.

[%FILE_NAMES] – This is the constant used along with 'MISSING_FILE_DESC' and 'MISSING_FILE_NAME' output fields and will have list of those file description that are missing in the current directory.

This can be used in creating an email content that requires those missing file names to be shared with email recipients.

BATCH_NUM – This is the constant having current batch number as a value. This constant can be used as a Rule output parameter value for any of the 'DRILL_KEY1/2/3' or 'EMAIL_SUBJECT' or 'EMAIL_BODY_CONTENT'.

FILE_NAME – This is the constant having input file name as a value. This constant can be used as a Rule output parameter value for any of the 'MESSAGE_PARM1/2/3/4/5' or 'EMAIL_SUBJECT' or 'EMAIL_BODY_CONTENT'.

10.3 Create Rule using Rule Type

1. Go to Admin Menu, select R and then click Rule. A sub-menu appears.
2. Click Add option from the Rule sub-menu. The Rule window appears. This window has following sections:
   - Main
   - Rule Output Parameters
   - Criteria

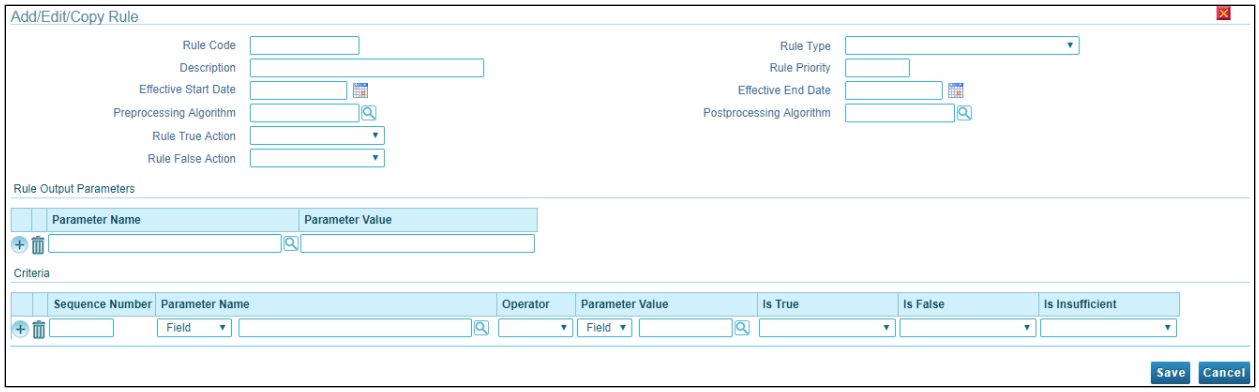


Figure 54: Rule

3. Enter unique **Rule Code** in corresponding field.
4. Select either of the Rule Type i.e. **File Upload Rule Failure Type (FILE_UP_FAIL)** or **File Upload Rule Type (FILE_RQ_TYPE)**.

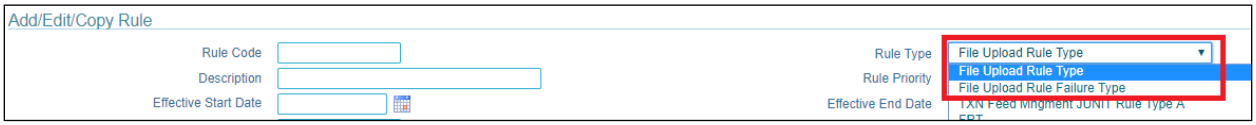


Figure 55: Rule

5. Enter **Rule Priority**, this will decide the execution **sequence** of this **Rule** for this **Rule Type.**
6. **Effective Start Date** should be less than Business Date and **Effective End Date** should not be blank and greater than **Effective Start Date.**
7. **Rule True Action** should be '**Success'** if we want to conclude with successful verification of this Rule. Else you can select either '**Next Dependent Rule**' or '**Next Rule by Priority**'.
8. **Rule False Action** can be either '**Next Dependent Rule**' or '**Next Rule by Priority**'.

10.3.1 Rule Output Parameters

1. You will have to configure required list of output parameters using the predefined fields of type '**Output**' corresponding to that of selected Rule Type i.e. **File Upload Rule Failure Type (FILE_UP_FAIL)** or **File Upload Rule Type (FILE_RQ_TYPE)**.
2. Rule Output Parameters are the fields with static data values.

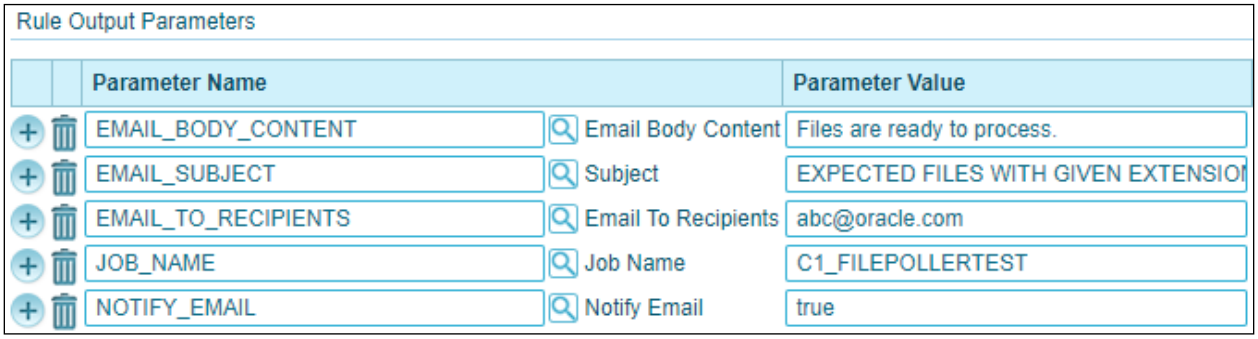


Figure 56: Rule output Parameters

3. This list of parameter names and values will be returned if the corresponding **Rule** is successful.

10.3.2 Criteria

1. You can define the criteria required for uploading the picked file into ORMB system.
2. Criteria is the conditions that are formed using **'Input'** type fields defined in corresponding Rule Type; i.e. either **File Upload Rule Failure Type (FILE_UP_FAIL)** or **File Upload Rule Type (FILE_RQ_TYPE)**.

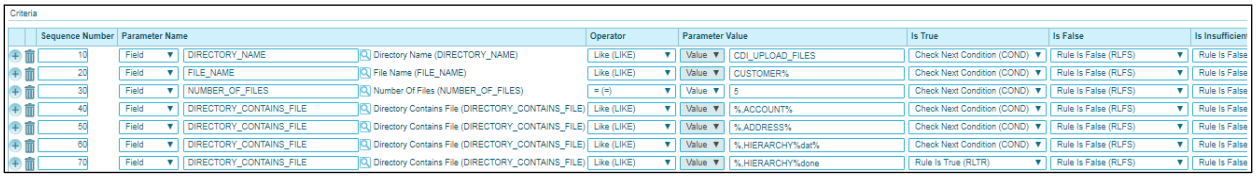


Figure 57: Criteria

3. Criteria are validated based on the list of input parameters with its corresponding values.
4. Input parameters are populated with in SFTP Poller batch and provided as input while invoking rule engine.