Oracle® Revenue Management and Billing

Version 2.7.0.1.0

File Upload Interface Batch Execution User Guide

Revision 2.0

F25012-01

November, 2019

ORACLE

Oracle Revenue Management and Billing File Upload Interface Batch Execution User Guide

F25012-01

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 detail information about various batches to be executed while performing tasks such as uploading, processing and updating status of files using File Upload Interface. It also lists and explains the parameters that you can specify while executing each File Upload batch.

Intended Audience

This document is intended for the following audience:

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

Organization of the Document

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

Section No.	Section Name	Description
Section 1	Introduction	Provides an overview of the File Upload Interface process. It also provides approaches for staging file details in ORMB system.
Section 2	Transforming and Uploading File	Lists and describes batch used for file transformation and upload
Section 3	Processing File Request	Lists and describes batch used for processing file requests
Section 4	Updating File Record Status	Lists and describes batch used for updating file record status

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.

Document	Description
File Upload Interface Quick Reference Guide	Describes parameters related to File Upload Interface Master Configuration and explains how to perform important tasks using File Upload Interface.
File Upload Interface User Guide	Lists and describes various features 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. Introduction

Oracle Revenue Management and Billing enables you to convert and integrate data on cloud. In other words, you can transform and upload files to ORMB staging. You can transfer data from any legacy system to ORMB. The system provides you with different batches, which can be executed using the File Upload Interface. These batches help you upload and process files and update their status.

Note: In this document, **record** represents a file "request" or "data line" or "payload" or "token".

ORMB provides following three batches that help you perform these sub-processes.

- Transforming and Uploading File (C1-FTRAN)
- Processing File Request (C1-FREQP)
- Updating File Record Status (C1-FRSUP)

Parameter Name	Description
C1-FTRAN	Used to upload either an XML file having records in compliance with ORMB service schema or transform and upload file in customer defined format
C1-FREQP	Used to process uploaded legacy file records in ORMB staging
C1-FRSUP	Used to update the status of bulk records from "ERROR" to "RETRY" status or "RETRY LIMIT EXCEED" to "RETRY" status or "PENDING" to "ERROR" status

There are two approaches for staging file details in ORMB system:

- **File upload without Transformation** –XML files that comply with its corresponding ORMB service schema alone are uploaded on SFTP server.
- **File upload with Transformation** – Files having records in client-supported format are uploaded on SFTP server. In this approach, before uploading into ORMB staging, all these file records should be transformed into XML, in compliance with its corresponding ORMB service schema.

For both these approaches, before uploading file records into system, batch performs the below listed validations:

- Validates file size (It should not be greater than 2 GB)
- If **File Encryption Flag** in master configuration is set to True, then decrypts the file.
- Validates, if **File Request Type** exists
- Validates, if File Request Type has **File Transform Flag** value set to True
- Validates file parsing
- If Header is mandatory, validates file for availability of `<header>` section
- Validates header details
- If Footer is mandatory, validates file for availability of `<footer>` section
- Validates footer details
- If **Validate Checksum** flag is set to True, then validates if `<FILE_NAME>.checksum` file is available on SFTP server

Note: Ensure that at least one record data is present in a file.

- Checks **File Atomicity**
  - If True, then this batch is executed using single transaction strategy. Here, if there is a single record failure, then all records are rolled back.
  - Else, this batch is executed using multithreaded strategy. Here, if there is a single record failure, then only that record is rolled back.

Checks Skip Duplicates

- If True, then this batch uploads records that were not uploaded earlier and mark their status as PENDING and for other remaining records or duplicates, mark status as SKIPPED.
- Else, batch uploads all the records in a file and mark their status as “PENDING”.

Note: If you change the File Request Type configuration to reflect the changes, you must flush the session by executing **F1-Flush** batch or restart the threadpool.

2. Transforming and Uploading File

The File Transformation and Upload batch (**C1-FTRAN**) is used to upload either an XML file having records in compliance with ORMB service schema, or transform and upload file in customer-defined format such as XML or CSV or PSV or JSON.

The records in a file are transformed into ORMB conform service schema and uploaded into ORMB system using "Record Transformation Algorithm". Every uploaded file details are logged with identifier (FILE_ID) in CI_FILE_REQUEST table with either PENDING or COMPLETE status. Note that the Record Transformation Algorithm is eligible only for XML, CSV, PSV, JSON, or Fixed Position file formats.

Note: If all records in a file are uploaded, then the file is uploaded with COMPLETE status. If any of the records in a file are not uploaded due to some failure, then the corresponding file is uploaded with PENDING status. This is to ensure restartability.

A file can have records of only one File Request Type and the batch execution is for a given File Request Type (Soft parameter). This batch is used to read files on SFTP server at the given file path 'Soft Parameter'. If the "File Validation Algorithm" is mapped to the corresponding File Request Type, then validation corresponding to header, footer and checksum can be performed. This batch reads one file at a time and tokenize all the records within that corresponding file. After reading and tokenizing all the file records, the algorithm creates a ThreadWorkUnit for every single token. Here, the number of records is equal to number of WorkUnits. Before logging file details and creating Thread WorkUnits, File Transformation batch performs the following validations:

- Validates File path: If validation fails, batch is terminated with status "ERROR" and the file details are not logged in CI_FILE_REQUEST table.
- Validates Error Log File path: If validation fails, batch is terminated with status "ERROR".
- Validates if the File exists on the provided file path: If validation fails, batch is terminated with status "ERROR" and the file details are not logged in CI_FILE_REQUEST table.
- Validates if the File Name with extension is same as that given in its File Request Type configuration. If validation fails, batch is terminated with status marked as ERROR and file details are not logged in CI_FILE_REQUEST table.
- Validates if the file size is not greater than 2 GB: If validation fails, batch is terminated with status COMPLETE. File details are logged with ERROR status and error logs in CI_FILE_REQUEST table.
- Validates if File Request Type is incorrect: If validation fails, batch is terminated with status marked as ERROR.
- Validates whether the file has a header section: If yes, the text present in the first line of the file is considered as a header. If file header does not exist, batch is terminated with status COMPLETE. File is entered with ERROR status and error details in CI_FILE_REQUEST table.
- Validates whether the file has a footer section: If yes, the text present in last line of a file will be considered as footer. If file footer does not exist, batch is terminated with status COMPLETE and the file is entered with ERROR status and error details in CI_FILE_REQUEST table.
- Validates if the file has at least one record: If yes, the number of records is equal to the number of lines in a file after excluding first (header) and last (footer) lines. If validation fails, batch is terminated with status COMPLETE and the file is entered with ERROR status and error details in CI_FILE_REQUEST table.

- Validates if the File already exists in ORMB system: If validation fails, batch is terminated with status marked as ERROR and the file is entered with REJECT status and its error details in CI_FILE_REQUEST table.

Note: This condition is valid only if **Validate Duplicate File Name** flag is True.

- For checksum validation, batch looks for the corresponding checksum file (. <FILE_NAME>.checksum) to get the particular checksum value. If validation fails, batch is terminated with status marked as ERROR and the file is entered with REJECT status and its error details in CI_FILE_REQUEST table.

Encrypted files are to be uploaded. If decryption fails, batch is terminated with status marked as COMPLETE and the file is entered with ERROR status and error details in CI_FILE_REQUEST table.

A list of Thread WorkUnits is created by tokenizing the number of records in that corresponding file. Every single token persists as a single file record, with unique REQUEST_ID identifier in CI_FILE_REQUEST_DETAIL table with PENDING status. All files on SFTP server having the same file extension, corresponding to that configured for the given file record type is read and uploaded in ORMB system.

There are two fields to capture Thread WorkUnit's corresponding XML (Payload) record data, either of which is populated:

- REQUEST with CLOB dataType – This is populated when record value with File Request Type having configuration "File Size Greater than 32KB" is set as TRUE.
- BO_DATA_AREA with VARCHAR2(32000) dataType – This is populated when request value with File Request Type having configuration "Record Size Greater than 32KB" is set as FALSE.

This handling is for better performance while uploading and processing file request data.

Note: The File Transformation and Upload (C1-FTRAN) batch will:

* Ignore the files that already exist in ORMB system

* Not Handle Java exceptions or unknown errors

This batch supports both single thread and multi thread execution using **File Atomicity** flag on File Request Type.

This batch supports two modes to create ThreadWorkUnits using errorLogFilePath (Execute batch using File Chunks) soft parameter. ThreadWorkUnits are required for batch execution.

- Fetch and create ThreadWorkUnits for all requests in PENDING or RETRY status for the given File Request Type from CI_FILE_REQUEST_DETAIL table. ThreadWorkUnit has supplemental data for following fields:

Field	Description
File ID	Used to indicate unique Identifier of a file
File Name	Used to indicate file name of the request object to be processed
Request Type	Used to indicate file request type of the request object to be processed
Request Object	Used to indicate payload or data line to be persist in ORMB system
File Retry	Used to identify if the batch is aborted with partial file requests processing
Duplicate	Used to skip request processing if there are multiple records with the same identifier in a same file
ID	Used to indicate identifier specific to Payment stage upload and Transaction business services
XPATH	Used to indicate algorithm specific to Payment stage upload and Transaction business services file request transformation
FIELD Name	Used to indicate field name specific to Payment stage upload and Transaction business services file request transformation

- Fetch and create ThreadWorkUnits for all requests in PENDING or RETRY status for the given File Request Type from `CI_FILE_REQUEST_DETAIL` table.
    - Number of files created is in proportionate with the number of threads. For example, if Number of Threads is 100, then 100 files are created.
    - These files are created at the same file path on SFTP server from where the process of reading the uploaded files is initiated.
    - All these newly created files have approximately equal number of requests. For example, if there are 1000 requests, then each file will have 10 requests or records.

  ThreadWorkUnit will have supplemental data only for following fields:

Field	Description
ID	Specific to Payment stage upload and Transaction business services
XPATH	Specific to Payment stage upload and Transaction business services file request transformation
FIELD Name	Specific to Payment stage upload and Transaction business services file request transformation

Note: The newly created temporary files are deleted after all the requests in that file are uploaded in ORMB system.

The File Transformation and Upload (C1-FTRAN) batch performs validation of different attributes present in:

- File Upload Interface Master Configuration
- File Request Type Configuration

2.1 File Upload Interface Master Configuration

The File Transformation and Upload batch (C1-FTRAN) validates following attributes in File Upload Interface Master Configuration:

- Validate Checksum:
  - If true, then checksum validation will be performed before reading and staging file details in ORMB system
  - If false, this flag skips this checksum validation.
- Audit Log Required:
  - If true, then individual file request status) transition will be logged in ORMB system. The request status can be:
    - PENDING
    - PROCESSED
    - ERROR
    - SKIPPED
    - INPROGRESS
    - RETRY
    - IGNORE
    - RETRYLIMITEXCEED
- Archive File:
  - If true, then file is moved to “Archive File Location” and in case of file validation failure, the error file is moved to “Archive Error File Location”
  - If false, then file is not archived
- File Encryption Required:
  - If true, then this batch gets the encrypted file on SFTP server and decrypt it using “File Decryption Algorithm”
  - If false, then encryption-decryption is not supported

For detailed information on these attributes, refer File Upload Interface Quick Reference Guide.

2.2 File Request Type Configuration

The File Transformation and Upload batch (C1-FTRAN) validates following attributes in File Request Type zone:

- **File Transformation Required:**
  - If True, then File request Transformation is performed using Request Transformation Algorithm that transforms either CSV or PSV or XML or JSON to ORMB conform service schema.
  - If false, then there is no transformation done to the file.
- **File Transformation Algorithm:** This algorithm is used for file request transformation into XML request in compliance with ORMB service schema. ORMB framework provides a sample

transformation algorithm C1-FRTA. If you want to use C1-FRTA algorithm, it is required to provide "Map Field XPath" under Data Transformation.

- **File Atomicity**:
  - If True, then the batch is executed in a single thread mode. This is applicable for **All or None transactions** Transaction Type.
  - If False, the batch is executed in a multi-thread mode. The multi-threading is based on the number of requests within the files. For multi-threaded batch, Standard Commit Strategy is used.
- **File Upload and Process:**
  - If True, in single thread mode, for a single record transaction failure every other record transaction done are rolled back. File status is marked as ERROR with its failure updates in `CI_FILE_REQUEST` table. All the corresponding file requests are updated with ERROR status in `CI_FILE_REQUEST_DETAIL` table.
  - If True, in multi-thread batch, for a single record transaction failure only that record transaction is rolled back. File status is marked as COMPLETE with its failure updates in `CI_FILE_REQUEST` table. Only the particular file requests is updated with ERROR status in `CI_FILE_REQUEST_DETAIL` table.
  - If False, then the requests in a file is uploaded in ORMB system with PENDING status.
- **Skip Duplicates**:
  - If True, then all the duplicate records within a corresponding file is skipped while uploading in ORMB staging.
  - If False, all the available records within a file is uploaded in ORMB staging.
- **Service Logs Required**:
  - If True, then service execution details corresponding to the individual request persist in `CI_FILE_REQUEST_DTL_SERVICE` table. For every individual request detail in this service table, you can capture its identifier details by configuring a "FK Reference" for the corresponding Service in a File Request Type. These identifier details are used to navigate to the entity specific user interface.
- **File Header Required**:
  - If True, then first row data is passed as header string to File Validation Algorithm.
  - If false, first row data is considered as another request data to be persisted in ORMB system.
- **Header XML Tag:**
  - This is required if File Format is **XML** and File Transformation Required attribute is set to **True** and File Header Required attribute is set to **True**. This has the name of header tag being used in XML.
- **File Footer Required**:
  - If True, then last row data is passed as header string to File Validation Algorithm.
  - If False, last row data is considered as another request data to be persisted in ORMB system.
- **Footer XML Tag:** This is required if File Format is **XML** and File Transform attribute is set to **True** and File Footer attribute is set to **True**. This has the name of footer tag being used in XML.
- **Root XML Tag**: This is required if File Format is **XML** and File Transform attribute is set to **True**. This will have the name of root tag being used in XML.

For detailed information on these attributes, refer to File Upload Interface Quick Reference Guide. You can specify the following soft parameters while executing this batch:

Field	Description	Mandatory (Yes / No)	Comments
File Path	Used to specify the relative path where you want to upload the file	Yes	The path always begins with either of the following: • **@SHARED_DIR** – Configured path of shared directory • **@INSTALL_DIR** – Configured path of installation directory, defined with property `spl.runtime.environ.SPLEBASE` in the spl.properties file
File Name	Used to specify the name of the file that you want to upload For example, CustomerOnboard	No	If left blank, all files which have file extension corresponding to its file request type are picked from the given file path. You can enter portion of a name, for example: **CUST%** picks filename ending with CUST **%CUST** picks filenames starting with CUST **%CUST%** picks filenames containing CUST
File Request Type	File is processed with reference to this File Request Type configuration and is uploaded in ORMB system against this File Request Type	Yes	
Error Log File Path	Used to specify the relative path where the CSV file with error logs are stored	Yes	The path will always begin with either of the following: • **@SHARED_DIR** – it is configured path of shared directory. • **@INSTALL_DIR** – it is configured path of installation directory, defined with property `spl.runtime.environ.SPLEBASE` in the spl.properties file
Execute batch using File Chunks	Used to decide the execution behavior of this batch If Y, then batch is executed using file chunks.	No	

Override maximum errors	Used to override maximum number of errors allowed before the 'Run' step is terminated	No	
Thread Pool Name	Used to specify the thread pool on which you want to execute the batch	No	

Note: If the File Transformation and Upload batch (C1- FTRAN) fails or aborts due to some reason, you can restart the batch again with the same set of parameters.

2.3 Post Execution Check/Clean Up

On successful completion of this batch, logged file details are updated with COMPLETE status. If "File Upload and Process" attribute value for corresponding File Request Type is set to **True**, then all requests in that file are processed and uploaded with its execution status. The execution status can be,

- PROCESSED
- ERROR
- SKIPPED
- INPROGRESS RETRY
- IGNORE
- RETRYLIMITEXCEED

3. Processing File Request (C1-FREQP)

The File Request Processing batch (C1-FREQP) is used to process uploaded legacy file records in ORMB staging. By default, records with PENDING and RETRY status are processed.

Note: Each File Request Processing batch (C1-FREQP) execution is done for a single File Request Type.

This batch gets all records with PENDING and RETRY status against the given File Request Type. For records with RETRY status, only those records are processed, for which RETRY count is either less than or equal to that of the configured RETRY count against its corresponding REQUEST TYPE. Before processing a file request, this batch validates:

- **Incorrect Error Log File path** - If this validation fails, batch is terminated with ERROR status.

Thread iteration strategy is used for this batch. A list of Thread WorkUnits is created on available requests with PENDING and RETRY for the given file request type in `CI_FILE_REQUEST_DETAIL` table. Each request is processed for all the configured services against its corresponding <FILE_REQUEST_TYPE>.

The batch performs following steps before processing requests:

Validates whether File Request Type exists: If validation fails,

- Error is logged in `CI_FILE_REQUEST_DTL_MSG` table against that specific request and status for this request in `CI_FILE_REQUEST_DETAIL` table is updated as ERROR

Validates whether 'Validate Request Payload' flag is **True**:

- If True, checks for the availability of XML payload for every individual service configured in the corresponding <FILE_REQUEST_TYPE>.
- If validation fails, the request is updated with ERROR status and error details are logged in `CI_FILE_REQUEST_DTL_MSG` table against this <REQUEST_ID>.

If pre-processing algorithm is configured for this service, then the same will be triggered. If Skip Flag is set to **True**, then this service execution is skipped and the Service status is updated as SKIPPED in `CI_FILE_REQUEST_DTL_SERVICE` table.

Parses the XML payload: If validation fails, Error is logged in `CI_FILE_REQUEST_DTL_MSG` table against that specific <REQUEST_ID> and status for this Request in `CI_FILE_REQUEST_DETAIL` table is updated as ERROR.

Service is executed for the given operation using the corresponding payload.

The batch performs following steps after request processing:

If execution is successful, then:

i. A record is created for that <SERVICE> in `CI_FILE_REQUEST_DTL_SERVICE` table.

ii. Against this service record, Primary key values are stamped only if this <SERVICE> has configuration of corresponding <FOREIGN_KEY_REF> in the <REQUEST_TYPE>.

iii. This Primary key values with its corresponding <FOREIGN_KEY_REF> is used for File Request dashboard for 360 degree view.

iv. Status of each service is:

- If Business Object life cycle status flag is configured True for that <SERVICE>, then its status will be INPROGRESS.
- If service execution is skipped, then its status will be SKIPPED.

- If there is a failure with error which is configured for <SENT_FOR_APPROVAL> in this <REQUEST> corresponding <REQUEST_TYPE>, then status will be SFAL.

After completing successful execution of this request with all its available services for that <REQUEST_TYPE>, status for this <REQUEST> will be updated as “PROCESSED”.

If execution fails, then

- Error is logged in `CI_FILE_REQUEST_DTL_MSG` table against that specific Request.
- Status for this Request in `CI_FILE_REQUEST_DETAIL` table will be:
  - If there is an error, which is configured for <RETRY> in this <REQUEST> corresponding <REQUEST_TYPE> then it will updated as RETRY.
  - If there is an error which is configured for <SENT_FOR_APPROVAL> in this <REQUEST> corresponding <REQUEST_TYPE> then status will be SFAL.
  - Else it will be updated as “ERROR”.
- No other remaining services are executed for this individual request.
- Every other service that was executed before will be rolled back.

Note:

* Service name is always the root element of its respective XML payload.

* Execution of services are done in a sequence with respect to the given <SEQ_NUM> for each <SERVICE>.

* Do not include <version> element in any of the payloads.

* Java exceptions or unknown errors are not handled.

A common API is exposed to update the <SERVICE> status. The status values may be either INPROGRESS or SFAL. After completion of respective service lifecycle or workflow, this API updates the status with the currently available final status. The status can be:

- If successful then PROCESSED
- If cancelled then CANCELLED
- If rejected then REJECT

Nesting is also supported in XML payloads. For two different configured <SERVICE> in a single <FILE_REQUEST_TYPE>, an XML payload can be provided with parent-child relationship. This can be achieved by “Dependent Service Name” attribute value.

There are two approaches in providing XML payloads:

- **Provide Dependent Service Name**: You can provide foreign Key reference attribute value in the “Dependent Schema Name” field. For example: <C1_PERSON_BO>
- No Dependent Service Name is provided

A sample XML payload when foreign Key reference attribute is provided is as follows. It defines a Person-Account relationship.

```
<root>
<request>
<requestType>CUSTONBOARD</requestType>
<payload>
<C1_PERSON_BO>
        <personOrBusiness>P</personOrBusiness>
        <division>CA</division>
        <personName>
                <nameType>PRIM</nameType>
                <entityName>Smith, Jones</entityName>
                <isPrimaryName>true</isPrimaryName>
        </personName>
        <C1-AccountBO>
                <setUpDate>2001-01-01</setUpDate>
                <division>CA</division>
                <accountNumber>
                        <isPrimaryId>true</isPrimaryId>
                        <accountIdentifierType>C1_F_ANO</accountIdentifierType>
                        <accountNumber>NewAcc2</accountNumber>
                </accountNumber>
                <accountPerson>
                        <accountRelationshipType>MAIN</accountRelationshipType>
                        <isFinanciallyResponsible>true</isFinanciallyResponsible>
                        <isMainCustomer>true</isMainCustomer>
                </accountPerson>
        </C1-AccountBO>
</C1_PERSON_BO>
</payload>
</request>
</root>
```


Nesting XML payload has following advantages:

- No need of providing any <PERSON> reference in <ACCOUNT>. For example, <accountPerson> relationship is handled internally.
- Better performance - It is not required to have a separate query to get parent primary key for mapping in child payload.

A sample XML payload where no “Dependent Service Name” attribute is provided is listed below:

```
<request>
<requestType>CUSTONBOARD</requestType>
<payload>
<C1_PERSON_BO>
        <personOrBusiness>P</personOrBusiness>
        <division>CA</division>
        <personName>
                <nameType>PRIM</nameType>
                <entityName>Smith, Jones</entityName>
                <isPrimaryName>true</isPrimaryName>
        </personName>
</C1_PERSON_BO>
</payload>
<payload>
<C1-AccountBO>
        <setUpDate>2001-01-01</setUpDate>
        <division>CA</division>
        <accountNumber>
                <isPrimaryId>true</isPrimaryId>
                <accountIdentifierType>C1_F_ANO</accountIdentifierType>
                <accountNumber>NewAcc2</accountNumber>
        </accountNumber>
        <accountPerson>
                <accountRelationshipType>MAIN</accountRelationshipType>
                <isFinanciallyResponsible>true</isFinanciallyResponsible>
                <isMainCustomer>true</isMainCustomer>
        </accountPerson>
</C1-AccountBO>
</payload>
</request>
```


This is a multi-threaded batch. The multi-threading is based on number of requests with PENDING and RETRY status available in staging `CI_FILE_REQUEST_DETAIL` table for the provided File Request Type. You can specify the following parameters while executing this batch:

Field	Description	Mandatory (Yes / No)	Comments
File Request Type	Used to specify records with status as either PENDING or "RETRY" for processing.	Yes	
Error Log File Path	Used to specify the relative path where the CSV file with error logs are stored	Yes	The path will always begin with either of the following: • **@SHARED_DIR** – it is configured path of shared directory. • **@INSTALL_DIR** – it is configured path of installation directory, defined with property `spl.runtime.environ.SPLEBASE` in the spl.properties file
File Request Status	Used when you want to process records that are in a particular status. The input status can be: PENDING RETRY	No	
File Name	Used to specify the name of the file that you want to upload For example, CustomerOnboard	No	If left blank, all files which have file extension corresponding to its file request type are picked from the given file path. You can enter portion of a name, for example: **CUST%** picks filename ending with CUST **%CUST** picks filenames starting with CUST **%CUST%** picks filenames containing CUST
File Request Upload Date	Used when you want to process records which were uploaded on a particular date	No	**Note**: You must specify the date in the YYYY-MM-DD format.
Override maximum errors	Used to override maximum number of errors allowed before the 'Run' step is terminated	No	
Thread Pool Name	Used to specify the thread pool on which you want to execute the batch	No	

Note: If the File Request Processing (C1-FREQP) batch fails or aborts due to some reason, you can restart the batch again with the same set of parameters.

3.1 Post Execution Check/Clean Up

On successful completion of this batch, File details are inserted with COMPLETE status and File Request Details will be inserted with PENDING status.

Note: If there is any change in <REQUEST_TYPE> configuration, then it can be replicated by either restarting the thread pool or executing Flush All Caches batch (F1-Flush).

4. Updating File Record Status (C1-FRSUP)

The File Record Status Update batch (C1-FRSUP) is used to update the status of bulk records from:

- ERROR to RETRY status or
- RETRY LIMIT EXCEED to RETRY status or
- PENDING to ERROR status

Once the file status is set to RETRY status, the file can be re-executed using File Request Processing batch (C1-FREQP). This batch is a multi-threaded batch. The multi-threading is based on number of requests with ERROR status.

You can specify the following parameters while executing this batch:

Field	Description	Mandatory (Yes / No)	Comments
File Name	Used to specify the name of the file for which the status needs to be updated	Yes	You can enter portion of a name, for example: **CUST%** picks filename ending with CUST **%CUST** picks filenames starting with CUST **%CUST%** picks filenames containing CUST
Status Update Reason	Used to indicate the reason why status is to be updated	Yes	
Record Status Code to be updated	Used to specify status codes File records with the specified status will be updated: • ERROR • Retry Limit Exceeded • PENDING	Yes	**Note:** By default, records with ERROR are updated.
File Request Type	Used to specify file request type for which the status will be updated	No	
Message Category	Used to specify File Request Error Message Category for which the status is updated	No	
Message Number	Used to specify File Request Error Message Number for which the status is updated	No	
External System	Used to specify the external system for which the records status is updated	No	

Override maximum errors	Used to override maximum number of errors allowed before the 'Run' step is terminated	No	
Thread Pool Name	Used to specify the thread pool on which you want to execute the batch	No	

Note: If the File Record Status Update batch (C1-FRSUP) fails or aborts due to some reason, you can restart the batch again with the same set of parameters.

4.1 Post Execution Check/Clean Up

On successful completion of this batch, the file request status is updated to RETRY which can be further re-processed.

5. SFTP File Upload Poller (C1-FPOLR)

The SFTP File Upload Poller batch (C1-FPOLR) is used to poll the files uploaded on SFTP server and trigger its corresponding batch job using Rule Engine configuration and corresponding DBMS scheduler configuration setup.

Note: SFTP File Upload Poller batch (C1-FPOLR) execution can be setup for a number of intervals using DBMS scheduler setup.

This batch reads all those uploaded files in root directory i.e. 'File Upload Directory' mentioned in 'File Upload Interface master configuration' on SFTP server. Before picking any file this batch validates:

- File Upload Root Directory configuration: If validation fails, batch will be terminated with ERROR status.
- File Upload Root Directory path should always start with either **@SHARED_DIR** or **@INSTALL_DIR**

If validation fails, batch is terminated with ERROR status.

Single Transaction strategy is being used for this batch. Only one Thread WorkUnit will be created to pick file(s) from SFTP server.

The batch performs following steps before reading file(s) to upload:

- Validates whether File Root Directory is mentioned in File Upload Interface configuration. If validation fails, batch is terminated in ERROR status with validation message "File Root Directory is not mentioned in File Upload Interface configuration".
- Load 'FileRequestTypeCache' instance i.e. cache File Upload Interface master configuration and all those available File Request Type(s) in ORMB system.
- Get all those supported file extension(s) for file upload with respect to available File Request Type(s).
- Get actual Directory path using the given logical path i.e. **@SHARED_DIR** or **@INSTALL_DIR**.
- Read and get list of files available with in this directory and its sub-directories.
  - If there is no file in this directory, invoke Rule Engine for Rule Type 'FILE_UP_FAIL'.
  - Else, loop this list of files to create:
    i. A list of file instances with its corresponding details like: File Name, File Size, File Extension, Directory Name, File Path, File Upload Date Time
    ii. A single directory instance with details:
      - File Name String (Comma separated list of files in this directory)
      - Number of files
      - Number of supported files (get this using step 3)
      - Number of non-supported files (get this using step 3)
- Iterate list of file instances to execute steps,
  - Invoke a rule engine for Rule Type 'FILE_RQ_TYPE'.
    i. If one of the rule is satisfied, then, Output parameter 'Job Name (JOB_NAME)' value is used to trigger DBMS Scheduler Job.
    ii. Else, in case if no rule is satisfied, rule engine for Rule Type 'FILE_UP_FAIL' is invoked, and an email can be sent to notify rule failure for that directory or file.

Note: Rules have to be defined for rule type 'FILE_RQ_TYPE'.

It is mandatory to have output parameter 'Job Name (JOB_NAME)' configured for each Rule that will is invoked for Rule Type 'FILE_RQ_TYPE'. There has to be a Scheduler Job setup available for this 'Job Name (JOB_NAME)'.

If you want to notify failure case, then you need to define rules for rule type 'FILE_UP_FAIL'. To define rule, navigate to Admin Menu → R → Rule

This batch is a single-threaded batch.

You can specify the following parameters while executing this batch:

Field	Description	Mandatory (Yes / No)	Default Value
Rule Type	Used to process rules to get the required job name to be executed	Yes	FILE_RQ_TYPE
Directory Path	A relative directory path to get the uploaded files The path will always begin with either of the following: • **@SHARED_DIR** – it is configured path of shared directory. **@INSTALL_DIR** – it is configured path of installation directory, defined with property `spl.runtime.environ.SPLEBASE` in the spl.properties file		
Rule Type for Failure Notification	Used to invoke the rules required for mail notification in failure conditions		FILE_UP_FAIL
Poller Log Enable Flag	Used to log the file poller details if the rule is successful Values allowed are either **Y** or **N**		N
Override maximum errors	Used to override maximum number of errors allowed before the 'Run' step is terminated		
Thread Pool Name	Used to specify the thread pool on which you want to execute the batch		

Note: If the SFTP File Upload Poller batch (C1-FPOLR) fails or aborts due to some reason, you can restart the batch again with the same set of parameters.

5.1 Post Execution Check/Clean Up

On successful completion of this batch, batch job can be triggered for a file upload.

Note: If there is any change in <RULE CONFIGURATION> configuration, then it can be replicated by either restarting the thread pool or executing Flush All Caches batch (F1-Flush).