Oracle® Revenue Management and Billing

Version 2.6.0.1.0

File Upload Interface (FUI) - Batch Execution Guide

Revision 1.0

E96821-01

May, 2018

ORACLE

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

E96821-01

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. 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
- Implementation Team
- Consulting Team
- Development Team

Organization of the Document

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

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

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

Contents

1. Introduction

Oracle Revenue Management and Billing enables you to convert and integrate data on cloud. In other words, you are able to 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 also update their status.

File Upload Interface (FUI) will be used for data conversion and data integration on cloud. The interface provides ability to transform and upload files to ORMB staging. It can also be used to transfer data from any Legacy system to ORMB.

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

1. Transforming and Uploading File (C1-FTRAN)
2. Processing File Request (C1-FREQP)
3. 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:

1. File upload without Transformation – The XML files that are in compliance with its corresponding ORMB service schema will only be uploaded on SFTP server.
2. File upload with Transformation – Files having records in client supported format will be uploaded on SFTP server. In this approach before uploading into ORMB staging, all these file records are required to be transformed into XML in compliance with its corresponding ORMB service schema.

For both the above mentioned approaches, before uploading file records into system, batch will perform below listed validations:

1. Validates file size. It should not be greater than 2 GB.
2. If file encryption flag in master configuration is set to True, then decrypts the file.
3. Validates if File Request Type exists.
4. Validates if File Request Type has File Transform Flag value set to True.
5. Validates file parsing.
6. If Header is mandatory, validates file for availability of `<header>` section.
7. Validates header details.

8. If Footer is mandatory, validates file for availability of `<footer>` section.
9. Validates footer details.
10. If "**Validate Checksum**" flag is set to "True". If 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.

11. Checks File Atomicity.
    - If True, then this batch will be executed using single transaction strategy. Here, if there is a single record failure, then every record will be rollbacked.
    - Else this batch will be executed using multithreaded strategy. Here, if there is a single record failure, then only that record will be rollbacked.
12. Checks Skip Duplicates.
    - If True, then this batch will upload records that were not uploaded before and mark their status as "PENDING" status and for other remaining records or duplicates mark their status as "SKIPPED".
    - Else, batch will upload 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.

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

2. Transforming and Uploading File (C1-FTRAN)

The **File Transformation and Upload (C1-FTRAN)** batch 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 will be transformed into ORMB conform service schema and uploaded into ORMB system using "Record Transformation Algorithm". Every uploaded file details will be logged with identifier (**FILE_ID)** in "**CI_FILE_REQUEST**" table with either "**PENDING**" or "**COMPLETE**" status. Note that the File Record Transformation is only eligible for **XML** or **CSV** or **PSV** or **JSON** or **Fixed Position** file formats.

Note: File will be uploaded with **"PENDING"** status only if all the records in a corresponding file are not uploaded due to some failure. Else file will be uploaded with **"COMPLETE"** status. This is to ensure restartability.

Each single file can have records for only one File Request Type.

This batch execution will be done 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 will read 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 will validate whether,

1. File path is valid. If validation fails batch will be terminated with status marked as "ERROR" and the file details are not logged in "**CI_FILE_REQUEST**" table.
2. Error Log File path is valid. If validation fails, batch will be terminated with status marked as "**ERROR**".
3. File exists on the provided file path. If validation fails, batch will be terminated with status marked as "**ERROR**" status and the file details are not logged in "**CI_FILE_REQUEST**" table.
4. File Name with extension is similar to that given in its File Request Type configuration. If validation fails, batch will be terminated with status marked as "ERROR" status and the file details are not logged in "**CI_FILE_REQUEST**" table.
5. File size is not greater than 2 GB. If validation fails, batch will be terminated with status marked as "**COMPLETE**", file details are logged with "ERROR" status and error logs in "**CI_FILE_REQUEST**" table.
6. File Request Type is incorrect. If validation fails, batch will be terminated with status marked as "ERROR".
7. Encrypted files are to be uploaded. If yes, file decryption will be done. If decryption fails, batch will be terminated with status marked as "**COMPLETE**" and the file will be entered with "**ERROR**" status and error details in "**CI_FILE_REQUEST**" table.

8. File has "**header**" section. The text present in first line of the file will be considered as a "**header**". If file header does not exists, batch will be terminated with status marked as "**COMPLETE**" and the file will be entered with "ERROR" status and error details in "**CI_FILE_REQUEST**" table.
9. File has "**footer**" section. The text present in last line of a file will be considered as"**footer**". If file footer does not exists, batch will be terminated with status marked as "**COMPLETE**" and the file will be entered with "**ERROR**" status and error details in "**CI_FILE_REQUEST**" table.
10. File has at least one "**record**". The number of "**records**" will be equal to number of lines in a file after excluding first (**header**) and last (**footer**) lines. If validation fails, batch will be terminated with status marked as "**COMPLETE**" and the file will be entered with "**ERROR**" status and error details in "**CI_FILE_REQUEST**" table.
11. File already exists in ORMB system. If validation fails, batch will be terminated with status marked as "**ERROR**" and the file will be 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".

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

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

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

1. **REQUEST with CLOB dataType** – This will be populated when record value with File Request Type having configuration "**File Size Greater than 32KB**" is set as **TRUE**.
2. **BO_DATA_AREA with VARCHAR2(32000) dataType** – This will be 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.

1. Fetch and create **ThreadWorkUnits** for all these number of requests in "**PENDING**" or "**RETRY**" status for the given "File Request Type" from "**CI_FILE_REQUEST_DETAIL**" table.

- ThreadWorkUnit will have 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 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.

2. Fetch and create **ThreadWorkUnits** for all these number of requests in "**PENDING**" or "**RETRY**" status for the given "**File Request Type**" from "**CI_FILE_REQUEST_DETAIL**" table.
    - Number of files will be created in proportionate to the number of threads. For example, if Number of Threads are 100, then 100 files will be created.
    - These files will be 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 will 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	This is specific to Payment stage upload and Transaction business services file request transformation.
FIELD Name	This is specific to Payment stage upload and Transaction business services file request transformation.

Note: The newly created temporary files will be 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

1. File Upload Interface Master Configuration
2. File Request Type Configuration

2.1 File Upload Interface Master Configuration

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

1. **"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.
2. "**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
3. **"Archive File"**
   - If true, then if there is no file validation failure then corresponding file will be moved to the provided "**Archive File Location**".
   - If false, error file will be moved to "**Archive Error File Location**".
4. **"File Encryption Required"**
   - If true, then this batch will get the encrypted file on SFTP server and decrypt it using this "**File Decryption Algorithm**".

For detailed information on the above mentioned attributes, refer *File Upload Interface Quick Reference Guide.*

2.2 File Request Type Configuration

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

1. **"File Transformation Required"**
   - If True, then File request Transformation will be performed using "**Request Transformation Algorithm**" that transforms either **CSV** or **PSV** or **XML** or **JSON** to ORMB conform service schema.
   - If false, the same provided payload within a file is used to invoke the corresponding service.
2. **"File Transformation Algorithm"**

- This algorithm will be 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 this algorithm, it is required to provide "**Map Field XPath**" for every configured field with "**File Segment** Type" as "**Field Detail**" in "**Transformation Details**" section on "**File Request Type**" configuration user interface.

3. **"File Atomicity"**
   - If True, then this batch will be executed in a single thread mode. This is applicable for "All or None transactions" Transaction Type.
   - If false, the batch will be executed in a multi-thread mode. The multi-threading is based on number of requests within the files. For multi-threaded batch, Standard Commit Strategy is used.
4. **"File Upload and Process"**
   - If True, in single thread mode, for a single record transaction failure every other record transaction done will be rollbacked. File status will be marked as "Error" with its failure updates in "CI_FILE_REQUEST" table. All the corresponding file requests will be 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 will be rollbacked. File status will be marked as "Complete" with its failure updates in "CI_FILE_REQUEST" table. Only the particular file requests will be updated with "Error" status in "CI_FILE_REQUEST_DETAIL" table.
   - If False, then the requests in a file will be uploaded in ORMB system with "**PENDING**" status.
5. **"Skip Duplicates"**
   - If True, then all the duplicate records within a corresponding file will be skipped while uploading in ORMB staging.
   - If False, all that available records within a file will be uploaded in ORMB staging.
6. **"Service Logs Required"**
   - If True, then service execution details corresponding to the individual request will 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 will be used to navigate to the entity specific user interface.
7. **"File Header Required"**
   - If True, then first row data will be passed as header string to "**File Validation Algorithm**".
   - If false, first row data will be considered as another request data to be persisted in ORMB system.
8. **"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 will have the name of header tag being used in XML.
9. **"File Footer Required"**

- If True, then last row data will be passed as header string to "**File Validation Algorithm**".
- If false, last row data will be considered as another request data to be persisted in ORMB system.

10. **"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 will have the name of footer tag being used in XML.
11. **"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 the above mentioned attributes, refer *File Upload Interface Quick Reference Guide*. You can specify the following soft parameters while executing this batch:

Parameter Name	Description	Mandatory (Yes or No)	Comments
File Path	Used to specify the relative path where you want to upload the file.	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 a 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 the files which have file extension corresponding to its file request type will be picked from the given file path. You can enter portion of a name. For example: • 'CUST%' - will pick filename ending with 'CUST'. • '%CUST' - will pick filenames which starting with 'CUST'. • '%CUST%' - will pick filenames containing 'CUST'.

Parameter Name	Description	Mandatory (Yes or No)	Comments
File Request Type	File will be processed with reference to this "**File Request Type**" configuration and will be 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 will be 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 a 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 will be executed using file chunks.	No	
Override maximum errors	Used to override the 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 (C1- FTRAN)** batch fails or aborts due to some reason, you can restart the batch again with the same set of parameters.

Post Execution Check/Clean Up

On successful completion of this batch, logged file details will be updated with "**COMPLETE**" status.

If "**File Upload and Process**" attribute value for corresponding "**File Request Type**" is set to **True**, then all those requests in that file will be 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 (C1-FREQP)** batch is used to process uploaded legacy file records in ORMB staging. By default records with "**PENDING**" and "**RETRY**" status will be processed.

Note: Each **File Request Processing (C1-FREQP)** batch execution will be done for a single **"File Request Type"**.

This batch will get all the records with **PENDING** and **RETRY** status against the given **File Request Type**. For records with **RETRY** status, only those records will be 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 any file request this batch will validate,

1. Incorrect Error Log File path - If validation fails, batch will be terminated with "**ERROR**" status.

Thread iteration strategy is being used for this batch. A list of Thread WorkUnits will be created on number of available requests with "**PENDING**" and "**RETRY**" for the given priority in "**CI_FILE_REQUEST_DETAIL**" table. Each request will be processed for all the configured services against its corresponding <FILE_REQUEST_TYPE>.

- The batch performs following steps before processing requests:

1. Validates whether File Request Type exists. If validation fails,
   a. Error will be logged in "**CI_FILE_REQUEST_DTL_MSG**" table against that specific Request and status for this Request in "**CI_FILE_REQUEST_DETAIL**" table will be updated as "**ERROR**".
2. 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,
   a. This request will be updated with "**ERROR**" status.
   b. Error details will be logged in "**CI_FILE_REQUEST_DTL_MSG**" table against this <REQUEST_ID>.
3. If pre-processing algorithm is configured for this service, then the same will be triggered.
   a. If Skip Flag is set to True, then this service execution will be skipped. The Service status is updated as "**SKIPPED**" in "**CI_FILE_REQUEST_DTL_SERVICE**" table.
4. Parses the XML payload. If validation fails,
   a. Error will be logged in "**CI_FILE_REQUEST_DTL_MSG**" table against that specific" **REQUEST_ID**" and status for this Request in "**CI_FILE_REQUEST_DETAIL**" table will be updated as "**ERROR**".

- Service will be executed for the given operation using the corresponding payload.
- The batch performs following steps after request processing:
    1. If execution is successful, then
        a. A record will be created for that <**SERVICE**> in "**CI_FILE_REQUEST_DTL_SERVICE**" table.
        b. Against this service record, Primary key values will be stamped only if this **<SERVICE>** has configuration of corresponding **<FOREIGN_KEY_REF>** in the **<REQUEST_TYPE>**.
        c. This Primary key values with its corresponding **<FOREIGN_KEY_REF>** will be used for File Request dashboard for 360 degree view.
        d. Status of each service will be,
            i. If Business Object life cycle status flag is configured True for that **<SERVICE>**, then its status will be "**INPROGRESS**".
            ii. If service execution skipped then status will be "**SKIPPED**".
            iii. 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**".
    2. If execution fails, then
        a. Error will be logged in "**CI_FILE_REQUEST_DTL_MSG**" table against that specific Request.
        b. 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**".
        a. No other remaining services will be executed for this individual request.
        b. Every other service that was executed before will be rollbacked.

Note: Service name will be always the root element of its respective **XML** payload.

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

Never include <version> element in any of the payloads.

Java exceptions or unknown errors will not be handled.

A common API will be 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 will update the status with the currently available final status. The status will be,

1. If successful then "**PROCESSED**".
2. If cancelled then "**CANCELLED**".
3. 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:

1. Provide Dependent Service Name - You can provide foreign Key reference attribute value in the "Dependent Schema Name" field. For example, <C1_PERSON_BO>.
2. 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:

1. No need of providing any <PERSON> reference in <ACCOUNT>. For example, <accountPerson> relationship will be handled internally.
2. 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 batch 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 "**PRIORITY**".

You can specify the following parameters while executing this batch:

Parameter Name	Description	Mandatory (Yes or 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 CSV file with error logs will be 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 a property "spl.runtime.environ.SPLEBASE" in the "spl.properties" file.
File Request Status	Used when you want process records which are in a particular status. The input status can be: • **PEN** • **RET**	No	

Parameter Name	Description	Mandatory (Yes or No)	Comments
File Name	Used to specify the name of the file that you want to process. For example, CustomerOnboard.	No	If left blank, all the files which have 'File Extension' corresponding to its File Request Type will be picked from the given file path. You can enter portion of a name. For example: • 'CUST%' will pick all files ending with 'CUST'. • '%CUST' will pick all files starting with 'CUST'. • '%CUST%' will pick files which contain 'CUST'.
File Request Upload Date	Used when you want to process records which were uploaded on a particular date. **Note:** You must specify the date in the YYYY-MM-DD format.	No	
Override maximum errors	Used to override the 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.

Post Execution Check/Clean Up

On successful completion of this batch, File details will be 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 either by restarting the thread pool or executing **Flush All Caches** (**F1-Flush**) batch.

4. Updating File Record Status (C1-FRSUP)

The **'File Record Status Update' (C1-FRSUP)** batch 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** (**C1-FREQP**) batch.

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:

Parameter Name	Description	Mandatory (Yes or 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%' will pick all files ending with 'CUST'. • '%CUST' will pick all files starting with 'CUST'. • '%CUST%' will pick files which contain '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. The file records with the specified status will be updated. Status allowed are: • ERR • RTLE • PEN **Note:** By default, records with "ERR" status will be updated.	Yes	
Request ID	Used to specify Request ID for which the status will be updated.	No	

Parameter Name	Description	Mandatory (Yes or No)	Comments
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 will be updated.	No	
Message Number	Used to specify File Request Error Message Number for which the status will be updated.	No	
External System	Used to specify the external system for which the records status will be updated.	No	
Override maximum errors	Used to override the maximum number of errors allowed before 'Run' process 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' (C1-FRSUP)** batch fails or aborts due to some reason, you can restart the batch again with the same set of parameters.

Post Execution Check/Clean Up

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