Oracle® Banking Digital Experience

File Upload Configuration Guide

Release 25.1.0.0.0
G43072-01
September 2025

Oracle Banking Digital Experience File Upload Configuration Guide, Release 25.1.0.0.0

G43072-01

Contents

Preface

Purpose i
Before you Begin i
Pre-requisites i
Audience i
Documentation Accessibility ii
Critical Patches ii
Diversity and Inclusion ii
Related Resources ii
Conventions ii
Screenshot Disclaimer iii
Acronyms and Abbreviations iii
Post-requisites iii

1 File Uploads

1.1 Using Enrichers in File Uploads 1

2 Reports

Index

Preface

- Purpose
- Before you Begin
- Pre-requisites
- Audience
- Documentation Accessibility
- Critical Patches
- Diversity and Inclusion
- Related Resources
- Conventions
- Screenshot Disclaimer
- Acronyms and Abbreviations
- Post-requisites

Purpose

This guide is designed to help acquaint you with the Oracle Banking application. This guide provides answers to specific features and procedures that the user need to be aware of the module to function successfully.

Before you Begin

Kindly refer to our **Getting Started User Guide** for common elements, including Symbols and Icons, Conventions Definitions, and so forth.

Pre-requisites

Specify **User ID** and **Password**, and login to **Home** screen.

Audience

This document is intended for the following audience:

- Customers
- Partners

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customer access to and use of Oracle support services will be pursuant to the terms and conditions specified in their Oracle order for the applicable services.

Critical Patches

Oracle advises customers to get all their security vulnerability information from the Oracle Critical Patch Update Advisory, which is available at Critical Patches, Security Alerts and Bulletins. All critical patches should be applied in a timely manner to ensure effective security, as strongly recommended by Oracle Software Security Assurance.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

Related Resources

For more information on any related features, refer to the following documents:

- Oracle Banking Digital Experience Installation Manuals
- Oracle Banking Digital Experience Licensing Manuals

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.

Convention	Meaning
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Screenshot Disclaimer

Personal information used in the interface or documents is dummy and does not exist in the real world. It is only for reference purposes; actual screens that appear in the application may vary based on selected browser, theme, and mobile devices.

Acronyms and Abbreviations

The list of the acronyms and abbreviations used in this guide are as follows:

Table 1 Acronyms and Abbreviations

Abbreviation	Description
OBDX	Oracle Banking Digital Experience

Post-requisites

After finishing all the requirements, please log out from the **Home** screen.

1

File Uploads

- Using Enrichers in File Uploads
  This topic provides information on **Using Enrichers in File Uploads**. **(For custom defined templates only, not required for out of box templates)**

1.1 Using Enrichers in File Uploads

This topic provides information on **Using Enrichers in File Uploads**. **(For custom defined templates only, not required for out of box templates)**

- Enrichers are used to enrich or fetch a value for a given field. Let's say the field is Debit Account Id and enricher is Account Currency, so it means that the currency for that debit account Id needs to be fetched or enriched.
- Enricher can have enricher arguments. These arguments are passed when the enricher is invoked.
- Enrichers are of 2 types
  - Upload File Enrichers
  - Static arguments (enricherArgs) – Value is passed directly from template to enricher as label string
  - Dynamic arguments (enricherDynArgs) – Value is derived from a previous field of the record
- Extract (Response) File Enrichers

How Enrichers are used in File Upload ?

- In File Upload XML template, the field which will enrich other fields must have 'enricher' attribute. This attribute must not be specified for the fields which would be enriched.
- The value of this `enricher` attribute is the `ENRICHMENT_ID`. To configure a new enricher we have to add an entry of the fully-qualified name of the new enricher in `METAINF\services\ com.ofss.digx.framework.fileupload.enrichment.IEnrichment` file. Currently OBDX support only Java enrichers.
- Enrichers can be in any package but must implement the `IEnrichment`' interface and should be annotated with `@Enricher(value = " {ENRICHMENT_ID}"), where ENRICHMENT_ID` is the id of the enricher used in template. Custom enrichers should also be annotated with @Custom. Custom enricher with the same `ENRICHMENT_ID` as of base will override the base enricher.

  Example
- Refer to the following figure of File Template : InternalFT.xml .

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<FileDefinition fileName="InternalFT"
    fileHandlerClassName="com.ofss.digx.app.fileupload.handlers.InternalFTFileHandler"
    decryptionClass="" charSet="UTF-8" delimiter="," comments=""
    isFirstRecHeader="false" simpleOrMixed="M" fillchar="" partialProcessing="100" transactionType="ITG">

    <RecordDefinition
        recordHandlerClassName="com.ofss.digx.app.fileupload.handlers.InternalFTRecHandler"
        recordType="B"
        dtoClassName="com.ofss.digx.domain.fileupload.entity.InternalFTDTO"
        multiplicity="-1" maxFields="10" comments=""
        parent="" length="" transaction="ITG"
        mixedIdentifier="A">
        <Field name="mixedIdentifier"/>
        <Field name="partyId"/>
        <Field name="debitAccountId" enricher="ACCTCURR" enricherArgs=""/>
        <Field name="amount" type="CD"/>
        <Field name="amountCurr"/>
        <Field name="valueDate" enricher="DATE" enricherArgs="dd-MM-yyyy"/>
        <Field name="creditAccountId" enricher="ACCTDETAILS"/>
        <Field name="debitNarrative"/>
        <Field name="creditNarrative"/>
        <Field name="purpose"/>
    </RecordDefinition>

    <RecordDefinition
        recordHandlerClassName="com.ofss.digx.app.fileupload.handlers.InternalFTRecHandler"
        recordType="B"
        dtoClassName="com.ofss.digx.domain.fileupload.entity.InternalFTBeneDTO"
        multiplicity="-1" maxFields="10" comments=""
        parent="" length="" transaction="ITGBEN"
        mixedIdentifier="B">
        <Field name="mixedIdentifier"/>
        <Field name="partyId"/>
        <Field name="debitAccountId" enricher="ACCTCURR" enricherArgs=""/>
        <Field name="amount" type="CD"/>
        <Field name="amountCurr"/>
        <Field name="valueDate" enricher="DATE" enricherArgs="dd-MM-yyyy"/>
        <Field name="beneId" enricher="BENE" enricherArgs="INTERNAL"/>
        <Field name="debitNarrative"/>
        <Field name="creditNarrative"/>
        <Field name="purpose"/>
    </RecordDefinition>
</FileDefinition>
```

Static Enrichers

- In above template, the field name `debitAccountId` has a enricher `ACCTCURR` with no enricherArgs. `ACCTCURR`' enrichment id would be looked for and `AccountCurrencyEnricher` class is invoked.
  This enricher derives the **`debitAccountCurr`**. Hence this attribute must be present in the record DTO with its setters defined.

```
6 usages
@Enricher(value = "ACCTCURR")
@Base
public class AccountCurrencyEnricher implements IEnrichment {

    /**
```

```
@Override
public HashMap<String, Object> enrich(HashMap<String, Object> parameters) throws Exception {

    SessionContext sessionContext = (SessionContext) ThreadAttribute.get(ThreadAttribute.SESSION_CONTEXT);
    FileUploadPolicyHelper policyHelper = FileUploadPolicyHelper.getInstance();
    policyHelper.fetchAccountId(sessionContext, new Account(parameters.get("value").toString()),
            parameters.get("fileRefId").toString());
    HashMap<String, Object> fields = new HashMap<String, Object>();
    String curr = policyHelper.fetchCurrencyForAccount(new Account(parameters.get("value").toString()),
            parameters.get("fileRefId").toString());
    fields.put("debitAccountCurr", curr == null ? "" : curr);
    fields.put("debitAccountId", parameters.get("value"));
    return fields;
}
```

- The field name **valueDate** has static enricherArgs `dd-MM-yyyy` meaning that the date has to be specifically in `dd-MM-yyyy` format. This value is simply available to the enricher for processing purpose. This enricher does not add any new field but simply modifies the value of the current field.

```
@Override
public HashMap<String, Object> enrich(HashMap<String, Object> parameters) throws Exception {
    DateFormat df = new SimpleDateFormat(parameters.get("enricherArgs").toString());
    Date date = null;
    HashMap<String, Object> fields = new HashMap<String, Object>();
    try {
        df.setLenient(false);
        date = df.parse(parameters.get("value").toString());
        fields.put(parameters.get("field").toString(), new com.ofss.fc.datatype.Date(date));
    } catch (ParseException e1) {
        Exception e = new Exception();
        e.setErrorCode(UploadErrorConstants.FU_INVALID_VALUE_DATE);
        throw e;
    }
    return fields;
```

Dynamic Enrichers

If `enricherDynArgs` is specified,

Example. `enricherDynArgs="beneId~beneName"` on beneficiary address field, the parser simply invokes getters on beneId and beneName fields and passes the values to the enricher in a map. It should be noted that these fields must be defined previously/above the beneficiary address field, so that parser has already completed the setter operation.

```
<Field name="beneId"/>
<Field name=" beneName"/>
<Field name="beneAddr" enricher="ADDRESSENRICHER" enricherDynArgs="
beneId~beneName "/>
```

Eg. Extract (Response) File Enrichers

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<FileDefinition fileType="csv" delimiter="," handler="com.ofss.digx.framework.fileupload.extract.CSVHandler" encryptionClass="">

   <RecordDefinition recordType="H">
      <Field name="record" label="RECORD" />
      <Field name="recRefId" label="RECORD REF NO" />
      <Field name="fileRefId" label="FILE REF NO" />
      <Field name="digxRefId" label="E-BANKING REF NO" />
      <Field name="contractRefId" label="CONTRACT REF NO" />
      <Field name="recStatus" label="RECORD STATUS" />
      <Field name="errCode" label="STATUS CODE" />
      <Field name="errMsg" label="STATUS DESCRIPTION" />
   </RecordDefinition>

   <RecordDefinition  query="ResponseList" recordType="B">
      <Field name="record" no="1" wrapchar='"'/>
      <Field name="recRefId" no="2"/>
      <Field name="fileRefId" no="3" />
      <Field name="digxRefId" no="4" />
      <Field name="contractRefId" no="5" />
      <Field name="recStatus" no="6" />
      <Field name="errCode" no="7"  enricher="ERRORMSG" enricherArgs=""/>
      <Field name="errMsg" no="8" />
   </RecordDefinition>

</FileDefinition>
```

Enrichers can be added to response file templates. The enricher class is invoked in the same way as upload templates. Example, in above case, localized error message need to be added to extracts from `errCode`. Extract enrichers do not support dynamic arguments

File Copy Configuration

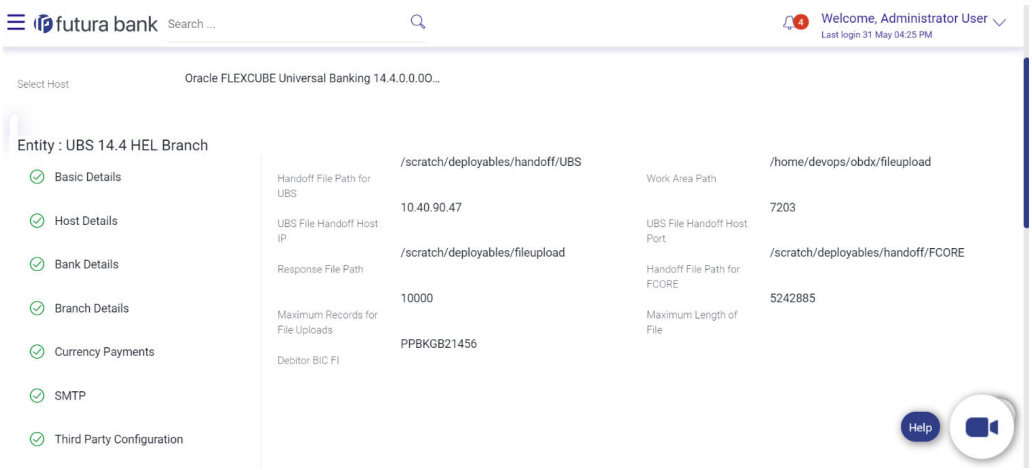
In case of OBPM as host, for **file level** uploads in OBDX, the files are generated in PAIN001001/PAIN001001 formats after approval at OBDX end is complete. These files are stored in a directory on OBDX server. For record level, service is used same as of single screen transactions.

Configs

1. Copying the file to host system

   File is copied via REST service to OBPM

2. Debtor BIC FI Configuration
   a. Provision to set Debtor BIC has been provided at entity level.
   b. The same can be configured in the following path by System Administrator user:
      i. Toggle menu → Configuration → System Configuration → Click on Continue → Select Entity → Dynamic Module Tab → File Upload



2

Reports

This topic provides information on **Reports**. Reports in OBDX can be used with Internal Reports Engine or Oracle BI.

Index

U

Using Enrichers in File Uploads, *1*