# File Upload Report Configuration Guide
# Oracle Banking Digital Experience
# Patchset Release 22.2.5.0.0

Part No. F72987-01

October 2024

ORACLE

File Upload Report Configuration Guide

October 2024

Oracle Financial Services Software Limited

Oracle Park

Off Western Express Highway

Goregaon (East)

Mumbai, Maharashtra 400 063

India

Worldwide Inquiries:

Phone: +91 22 6718 3000

Fax:+91 22 6718 3001

www.oracle.com/financialservices/



# Table of Contents

# 1. Preface

## 1.1 Purpose

Welcome to the User Guide for Oracle Banking Digital Experience. This guide explains the operations that the user will follow while using the application.

## 1.2 Audience

This manual is intended for Customers and Partners who setup and use Oracle Banking Digital Experience.

## 1.3 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 customers that have purchased support have access to electronic support through My Oracle Support. For information, visit, http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

## 1.4 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.

## 1.5 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.

## 1.6 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. |
| monospace | Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter. |

## 1.7 Screenshot Disclaimer

The images of screens used in this user manual are for illustrative purpose only, to provide improved understanding of the functionality; actual screens that appear in the application may vary based on selected browser, theme, and mobile devices.

## 1.8 Acronyms and Abbreviations

The list of the acronyms and abbreviations that you are likely to find in the manual are as follows:

| Abbreviation | Description |
|---|---|
| **OBDX** | Oracle Banking Digital Experience |

# 2. File Uploads

## 2.1 OutsideIn (For MS Excel processing)

Outside Inn - This is used for parsing XLS, XLSX in file uploads module. This library is not shipped with OBDX but needs to be downloaded from below link for required platform (OS on which app server is running)

http://www.oracle.com/technetwork/middleware/webcenter/content/oit-dl-otn-097435.html

Search Export – (Refer Pre requisite installation document for version)

Unzip the downloaded file and copy all contents of 'redist' folder to config/outsidein/<os> directory

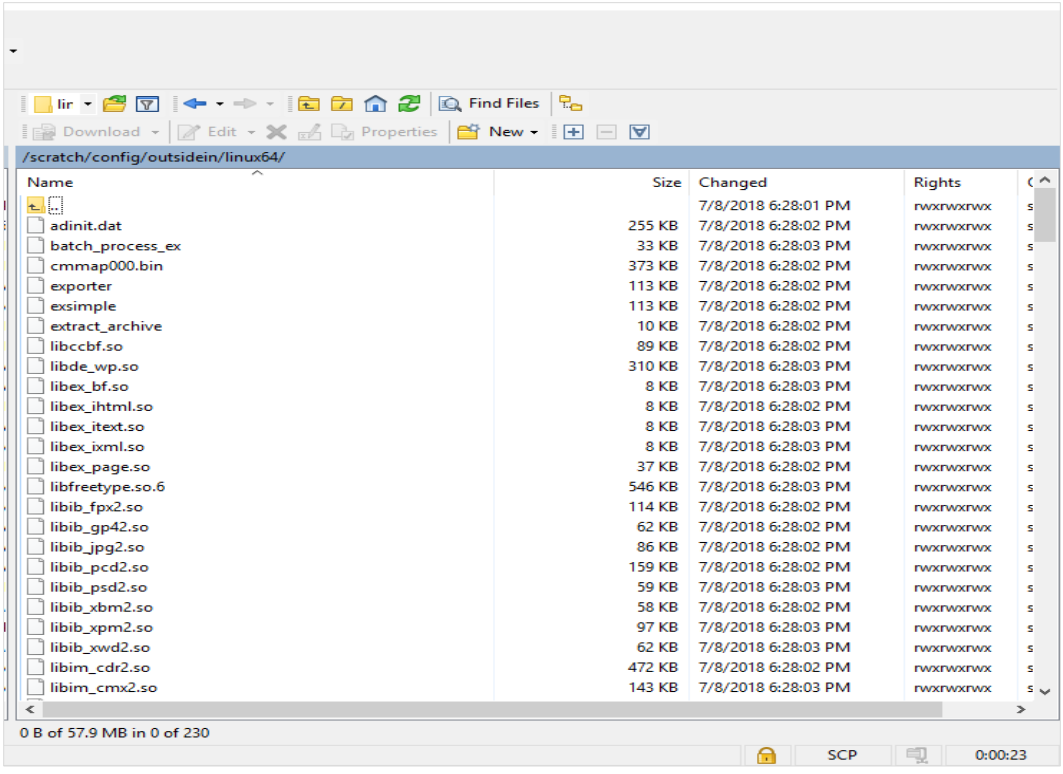
Then copy all contents (except jar & sh files) of 'sdk/demo' directory to config/outsidein/<os>

Use sx.cfg (replace/merge contents if required) shipped in installer from folder config/outsidein/<os>

Confirm/update path → select * from digx_fw_config_all_b where prop_id = 'OUTSIDE_IN_SDK'

Default config/outsidein/linux64

Grant 777 privileges for OutsideIn directory

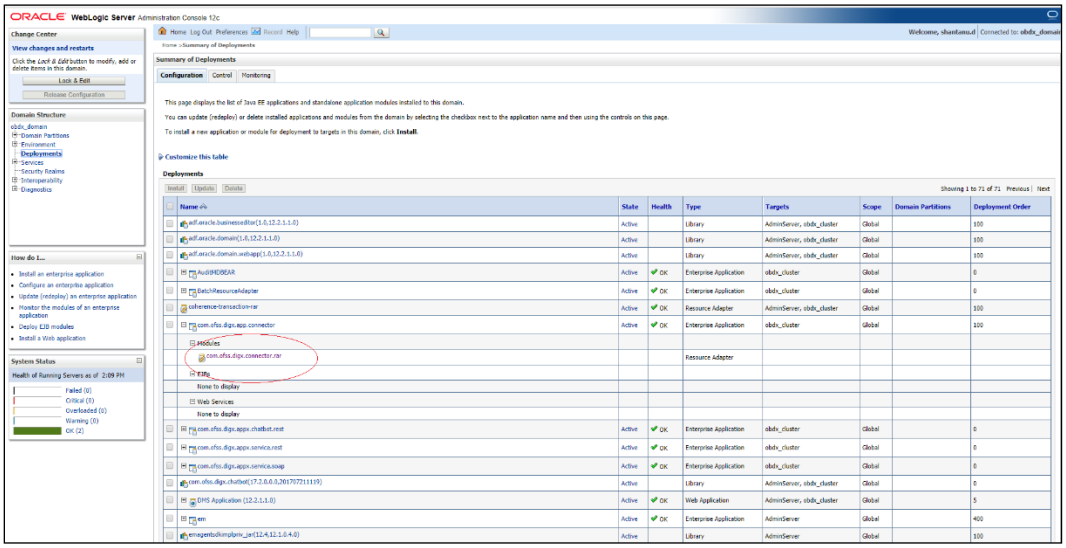


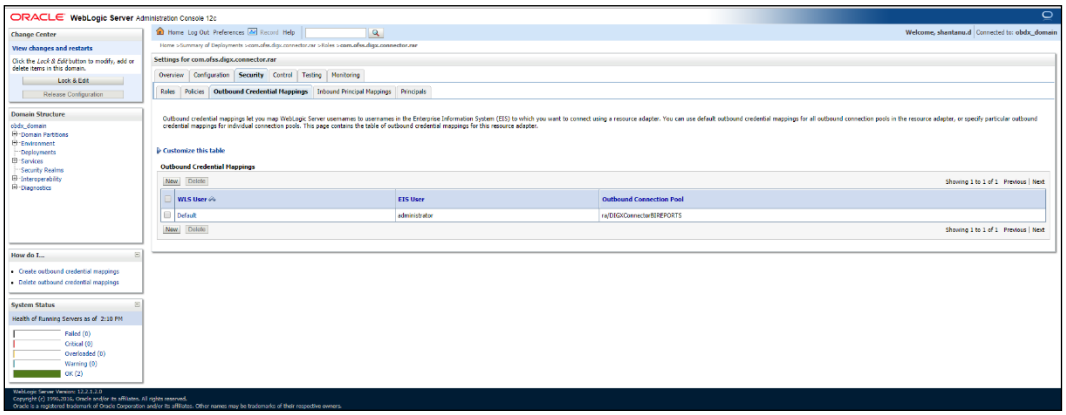
## 2.2 Configuration for storing key for decrypting uploaded files and creating encrypted response files

The key used for file decryption by default decryptor is stored in database in digx_fw_config_all_b with prop_id as ‘ENCRYPTION_KEY’. If this is to be stored in WLS connector update the property as below

update digx_fw_config_all_b set prop_value='KEY_STORE' where prop_id='ENCRYPTION_KEY_LOCATION';

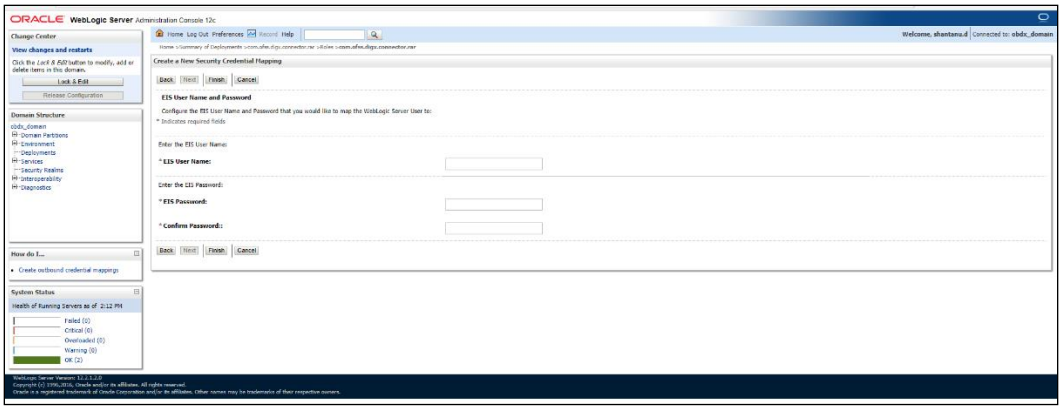
Update the encryption key in connector as below –





Click New > Select ra/DIGXConnectorFILEUPLOAD > Next > Select Default User

In password field, enter the encryption key




## 2.3 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' which is a column in table 'DIGX_FW_ENRICHMENTS_B'. Currently OBDX support only Java enrichers. Enrichers can be in any package but must implement the 'IEnrichment' interface.
- On the basis of the 'enricher' attribute value mapping is done from table 'DIGX_FW_ENRICHMENTS_B' and the corresponding 'ENRICHMENT_VALUE' column value is fetched and enrich() method of the specified Java class is invoked Eg.
- 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. In this case ‘DIGX_FW_ENRICHMENTS_B’ will be queried and search for ‘ACCTCURR’ and ‘AccountCurrencyEnricher’ class is invoked.

This enricher derives the debitAccountCurr. Hence this attribute must be present in the record DTO with its setters defined.

```
@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

Eg. 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. Eg, 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. Set this path as the value for prop_id = *‘UBS_HANDOFF_FILE_PATH’* in the *DIGX_FW_CONFIG_VAR_B* table against th`e required entity (Empty folder with full permission).

2. ***Copying the file to host system***

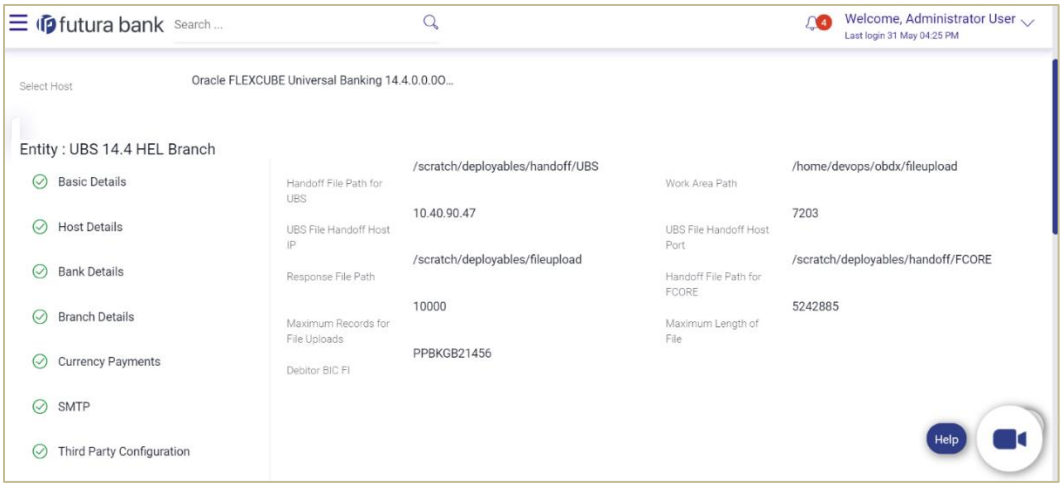
   *File is copied via REST service to OBPM*

3. 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*



# 3. Reports

**Reports in OBDX can be used with Internal Reports Engine or Oracle BI.**

## 3.1 Reports – Internal Report Engine

In installer scripts, all reports point to Internal report engine, no additional configuration is required.

**Note** : A8_C2_PENDING_APPROVALS works only with BI.

**For API Summary reports, internal engine works for maximum 500 records only**. For higher load BI is recommended.