ORACLE

Using External Data Integration Services for Oracle ERP Cloud

ORCALE WHITE PAPER | October 2023

Version [1.0]

Public

ORACLE

Purpose statement

This document provides an overview of features and enhancements for Oracle 'External Data Integration Service'. It is intended solely to help you assess the business benefits of Oracle 'External Data Integration Service' and planning for the implementation and upgrade of the product features described.

Disclaimer

This document in any form, software or printed matter, contains proprietary information that is the exclusive property of Oracle. Your access to and use of this confidential material is subject to the terms and conditions of your Oracle software license and service agreement, which has been executed and with which you agree to comply. This document and information contained herein may not be disclosed, copied, reproduced, or distributed to anyone outside Oracle without prior written consent of Oracle. This document is not part of your license agreement, nor can it be incorporated into any contractual agreement with Oracle or its subsidiaries or affiliates.

This document is for informational purposes only and is intended solely to assist you in planning for the implementation and upgrade of the product features described. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, timing, and pricing of any features or functionality described in this document remains at the sole discretion of Oracle. Due to the nature of the product architecture, it may not be possible to safely include all features described in this document without risking significant destabilization of the code.

ORACLE

Table of contents

ORACLE

Overview

Business organizations typically have a recurring need to streamline management of inbound and outbound data in areas such as initial data conversion, master data creation and maintenance, regular transaction processing, and fiduciary compliance. Oracle ERP Cloud offers a comprehensive collection of tools and features to meet these requirements. Oracle ERP integration scenarios generally involve system-to-system integration flows between distinct on-premises systems, third-party or legacy systems, and Cloud systems.

What's New

With Oracle ERP Cloud, these new features have been added to further simplify and enhance the Oracle ERP Integration Service capabilities:

- New operations, importBulkData and exportBulkData that further simplify bulk data management.
- Encryption option to secure data files for import and export processes.
- Capability to efficiently handle large files.
- Option to purge product application interface tables where needed.
- Multi-threading bulk import process.
- Option to select precise output file type for ExportBulkData, that further reduces file size and download time. See *Appendix 19: ExportBulkData Operation Precise ExtractFileType Support.*
- Multiple document ID support for the Load Interface File for Import job using ImportBulkData operation of ERP Integration Service. See *Appendix 12: Manual Inbound (Import) Steps.*

Inbound Data Management

Overview

There are several scenarios where data from on-premises or external business systems needs to be imported into Oracle ERP Cloud to accomplish business transactions such as:

- Recurring billing transactions originating from on-premises or PaaS-based applications which will be imported into Oracle ERP Cloud.
- Claims generated from on-premises insurance claim processing applications that require the creation of Payables invoices for remitting payments.
- Journal entries from legacy applications which will be imported into Oracle ERP Cloud.

See *Appendix 21: Supported Import Processes by Load Interface File for Import Job: Supported File based Data Import Processes details.*

Key Features

- Supports high-volume data import scenarios
- Support of legacy data migration, as well as recurring bulk data import
- Automation of end-to-end import flows with web service architecture
- Tracking of import processes for completion, errors, and resubmission
- Notifications in the form of emails, bell notification, business events and web-service based call-back to automate data validation and error resolution
- Predefined import templates for business objects

Details

External data integration services for accommodating inbound data in Oracle ERP Cloud include the following components:

- Templates to structure, format, and generate the data file according to the requirements of the target application objects.
- File-based load process to load the data file(s) into the respective product application interface tables.
- Application-specific data import processes to transfer data from product application interface tables to the relevant product application tables.

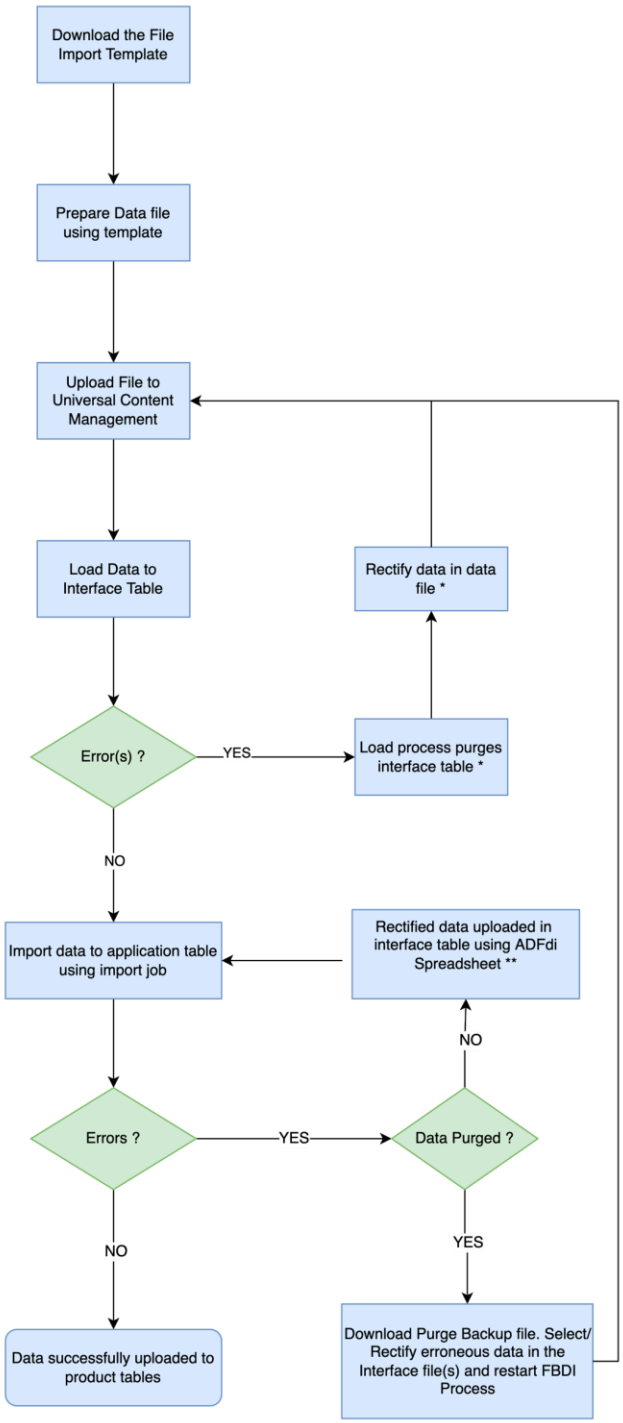


Figure 1: External data integration conceptual process flow diagram

* In case of error while loading data to the interface table, it is recommended to purge data from the interface table and reload the data after rectifications. However, the error handling strategies might differ for different import processes.

** The data rectification tool will differ for different import process.

Automated End-to-End Inbound (Bulk Import) Orchestrated Flow

To automatically import data into Oracle ERP Cloud:

- Create the data file using the applicable inbound business object template. See *Appendix 17: Steps to get list of all supported Import Processes for Load Interface File for Import job.*
- Invoke the Oracle ERP Integration Service to initiate import.
- Provide notification through asynchronous callback upon completion.
- Deliver the import status and information using web service-based callback or business events to access logs or errors.
- Review any errors if applicable and take appropriate action for error resolution.

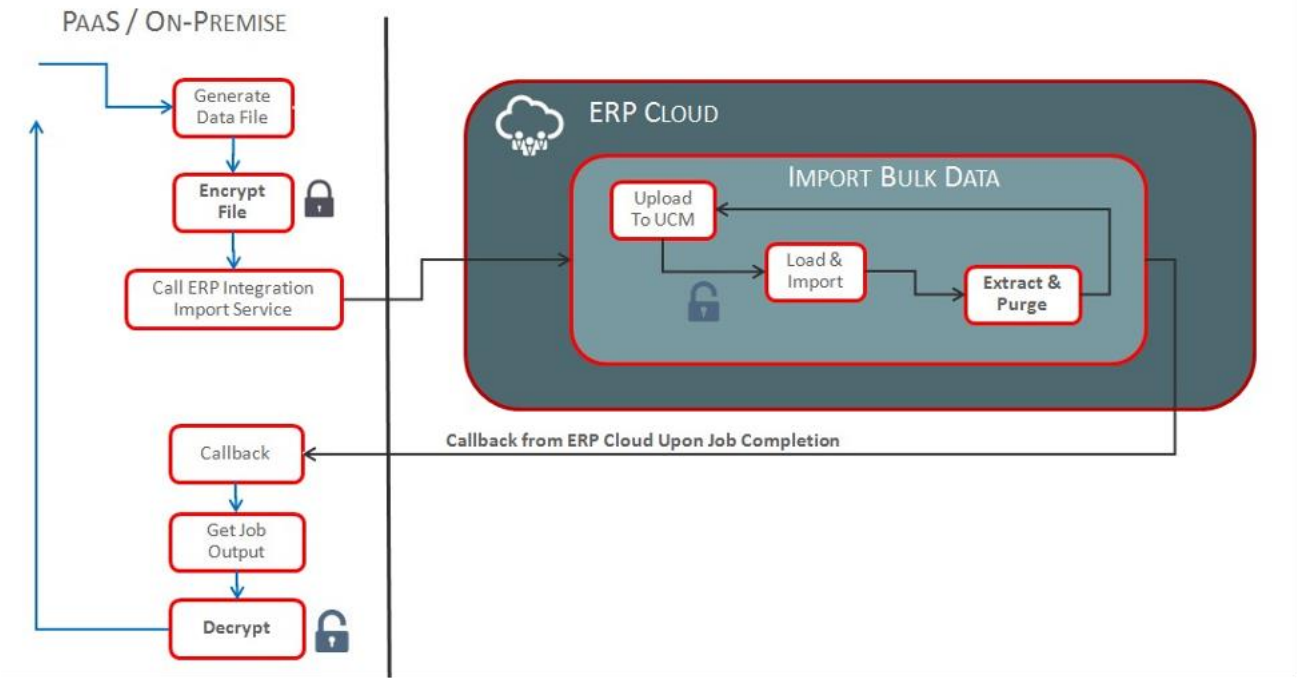


Figure 2: Inbound data integration orchestration flow

Note: After completion, Oracle ERP Cloud extracts data from the interface and error tables, includes the ESS job log files, and uploads the files to the UCM server in a ZIP file format. Once uploaded successfully to the respective UCM account, data from the interface and error tables will be purged.

Example

To illustrate the inbound data integration with Oracle ERP Cloud, the Journals Import flow will be used as an example. A batch of journal entries will be interfaced to application with help data created by user in a supported file format.

Prerequisites

- Import a certificate into your local keystore. For more information, see *Appendix 5: Testing Web Service using a Client Proxy.*
- Configure the sample web service tester. For more information, see *Appendix 5: Testing Web Service using a Client Proxy.*
- Identify the user name and password to call the Import Journals process.
- Verify that the user has access to the 'AttachmentsRead 'role in the UCM server. For more information, see *Appendix 1: Security Prerequisites to Download the Job Output File.*
- Verify the end point URL for the web service. For more information, see the automation web service at
- *https://<hostname>.<domainname>/fscmService/ErpIntegrationService?WSDL.*

Flow Steps

1. Generate the data file for the object you want to import. For more information, see Generating the Inbound Data File.
2. Prepare the request payload for the ImportBulkData operation of the Oracle ERP Integration Service. This web service operation performs the following tasks:
    a. Uploads the data file to the UCM server.
    b. Loads data from the file on the UCM server to the respective product interface table(s).
    c. Imports the data from the product interface table(s) to the respective Oracle ERP product main table(s).
    d. Extracts errors and logs into a ZIP file and uploads them to the respective UCM account.
    e. Purges the interface and errors tables related to the respective import job.
    f. Notifies users upon completion of all ESS jobs using bell, email, or callback URL as defined in the payload.
3. Receive a bell, email, or web service bases callback or business event notification for the request identifier returned by the web service operation in step 2.
4. Prepare the payload for the 'getDocumentForDocumentId' operation to download the output file.

Generating the Inbound Data File

The File-Based Data Import guide available on the Oracle Help Center (http://docs.oracle.com) include integration templates to help you prepare external data for loading and importing. Each template includes table-specific instructions, guidelines, formatted spreadsheets, and best practices for preparing the data file for upload. Use the templates to ensure that your data confirms to the structure and format of the target application tables.

When preparing external data using the templates for import, you need to complete the following tasks:

- Download the applicable import template.
- Prepare data using the correct spreadsheet import template.

Downloading a Template

1. Open the File-Based Data Import guide for your cloud service. Locate the import process of interest.
    a. FBDI Templates for Financials: https://docs.oracle.com/pls/topic/lookup?ctx=fa-latest&id=OEFBF
    b. FBDI templates for Projects: https://docs.oracle.com/pls/topic/lookup?ctx=fa-latest&id=OEFPP
    c. FBDI Template for SCM: https://docs.oracle.com/pls/topic/lookup?ctx=fa-latest&id=OEFSC
    d. FBDI Template for Procurement: https://docs.oracle.com/pls/topic/lookup?ctx=fa-latest&id=OEFBP
2. View the list of files:
    a. Control files describe the logical flow of the data load process.
    b. Spreadsheet templates include the worksheets and macros for structuring, formatting, and generating your data file.

3. Click the applicable template URL in the File Links table to download the file. For example, click JournalImportTemplate.xlsm in the Journal Import topic.

Preparing Data Using the Spreadsheet Template

Steps to prepare your data in a spreadsheet format:

1. Open the spreadsheet template. The first worksheet in each file provides instructions for using the template.

   Important: Follow the instructions otherwise you may get data load errors and data import failures. If the file is machine generated, you must use the UTF-8 encoding to avoid load errors.

2. Enter the required data and save the file.
3. Click Generate CSV File.

The macro generates a comma-separated values (CSV) file and compresses the file into a ZIP file. You must transfer the ZIP file to the Oracle Content Management Server (UCM).

Overview of Template Structure

The integration templates include the following characteristics:

- Each interface table is represented by a separate worksheet.
- Each interface table field is represented by a worksheet column with a header in the first row.
- Each column header contains bubble help text or help comments that include details about the column such as the expected data type, length, and in some cases, other relevant instruction text.
- Columns are formatted where applicable to match the target field data type to eliminate data entry errors.
- The worksheet columns are in the order that the control file processes the data file.

For more information on the template structure, see the main Instructions worksheet in the template.

Template Requirements

To minimize the risks of an unsuccessful data load, ensure these are followed:

- Unused columns can be hidden, but they cannot be reordered or deleted.

  Important: Deleting or reordering columns causes the load process to fail and results in an unsuccessful data load.

- External data must conform to the data types accepted by the control file and process for the associated database column.
- Date column values must appear in the YYYY/MM/DD format.
- Amount column values can't have separators other than a period (.) as the decimal separator.
- Negative values must be preceded by the minus (-) sign.
- Column values that require whole numbers include data validation to allow whole numbers only.
- For columns that require internal ID values, refer to the bubble help text for additional guidance about finding these values. For examples Purchase Order ID or Invoice ID.
- The XLSM template needs to run on the OS with date format set to "English US ".

After you finish preparing the data in the applicable spreadsheet template worksheet(s), click the Generate CSV File button on the main Instructions worksheet to generate a ZIP file containing one or more CSV data files.

Caution

1. Recommendation is to generate the data file using the applicable Inbound business object template.
2. If the file is being generated by other means like Java programs, manual creation though any editor etc., then ensure that the data is represented in the structure and format of the target application tables.

3. Only *.txt, *.dat, *.csv, *.xml, and *.ack file formats or ZIP file compressed with a Java 8 supported compression method containing the mentioned accepted format file(s) are accepted. Best practice is to use ZIP files instead of raw files for better performance.
4. Recommended compression method is DEFLATED.
5. Only UTF-8 encoded files should be used.

Error Correction

This section talks about rectifications of error occurred during various phase of File import.

Correcting Load Process Errors

The Load Interface File for Import process ends in error when the load of the data file fails for any individual row. The Load File to Interface child process ends as an error or warning. All rows that were loaded by the process are deleted and the entire batch of records is rejected.

Correcting Interface Data Errors

Steps to correct errors:

1. Review the upload error logs.
2. Change any structural or formatting anomalies in the data.
3. Generate the ZIP file containing the CSV files using the respective import template.
4. Upload the corrected file to the UCM server and resubmit the Load Interface File for Import process.
5. Repeat these steps until the process successfully loads all the data.

Correcting Import Process Errors

If the import process fails with errors:

1. Review the errors in the import log (See *Operation: downloadESSJobExecutionDetails for detailed steps).*
2. Correct the error records using the applicable ADFdi correction spreadsheets.

   For a list of import processes and their corresponding ADFdi correction spreadsheets, see *Appendix 7: Error Handling for Import Jobs.*

   If auto purge is enabled in your import process, then you cannot use ADFdi. Use these steps instead:

   1. Download the purge erroneous ZIP file from the File Import and Export page.
   2. Select the erroneous data records from the interface file and correct them.
   3. Follow the FBDI process to resubmit the corrected data.

Purging Interface and Error Tables

Data from the interface and error tables can be purged as part of the following processes:

1. Few of the FBDI import process (Import Journal, Import Payable Invoice etc.) may purge successfully imported data from interface table based on the functional requirement.
2. Customers also have the option to manage the purge process directly from the Scheduled Processes page by launching the Purge Interface Tables ESS job, as needed. This ESS job supports the purge of interface data created from either FBDI or non-FBDI sources based on registered criteria in the seed data, see *Appendix 16: Purge - UI Based Approach.*
3. If the data is imported through the importBulkData operation of ERP Integration Service, then customer can pass 'purgeOption=Y' in jobOptions, which will trigger an instance of 'Purge Interface Tables Job' with FBDI intent for the corresponding load request ID.

For the FBDI intent purge, the purge backup file is stored and associated with the respective UCM import account for reference where needed. When the 'Purge Interface Tables Job' is triggered for FBDI intent, the user must explicitly opt-in to take the backup. The file can either be downloaded using the Oracle ERP Integration Service or

from the File Import and Export page. This file is a consolidated ZIP file that contains the individual interface and error data files in a comma separated values (CSV) format.

For data correction, select and revise any erroneous data from the respective interface spreadsheet file, then upload the revised interface file again to execute the FBDI process.

For the processes outlined above, the existing inbound, outbound, and erroneous data files older than 30 days stored on the UCM server, will automatically be purged for the applicable UCM account.

Operation: extractAndPurge:

The extractAndPurge operation uploads the load and import job output and logs to the Universal Content Management server and purge interface tables with web service-based callback and notification.

The purge file naming convention is as follows: ImportBulkData_<ImportJobName>_<LoadRequestId>.zip The following table lists the parameters for this operation:

Parameter Name	Description	Parameter (In/Out)	Mandatory	Type
Request IDs	The request ID(s) of load jobs.	IN	Yes	java.lang.String
Notification Code	A two-digit number that determines how and when a notification is passed for the status of the import job. See the table below for the notification code values.	IN	Yes	java.lang.String
Callback URL	The callback URL of the web service you implemented to receive the ESS job status upon job completion.	IN	No	java.lang.String
Job Options	There are no additional job options for this operation.	IN	No	java.lang.String

The following table provides information on the notification codes:

Digit Position	Digit Value	Meaning
First digit	1	Email notification

	2	Bell notification
	3	Email and bell notification
Second digit	0	Send in any case (import failed or succeeded)
	1	Send on import success
	2	Send on import failure

The following sample request payload illustrates the extractAndPurge process:

```
<soap:Body>
  <ns1:extractAndPurge
xmlns:ns1="http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationService/types/" >
    <ns1:requestIds>1234;1235;1236</ns1:requestIds>
    <ns1:notificationCode>30</ns1:notificationCode>
    <ns1:callbackURL>#NULL</ns1:callbackURL>
    <ns1:jobOptions></ns1:jobOptions>
</ns1:extractAndPurge>
</soap:Body>
```

Figure 3: Sample request payload for the extractAndPurge operation

To manage the purge process directly from the Scheduled Processes page by launching the Purge Interface Tables process as needed, see *Appendix 16: Purge - UI Based Approach.*

Outbound Data Management

Overview

Global statutory or fiduciary requirements drive diverse reporting and data extract needs. In these types of business scenarios, the flow of data from Oracle ERP Cloud is utilized for either 1) end-state reporting to internal business stakeholders, financial institutions, government agencies, tax authorities, or third parties or 2) as an intermediate means to perform additional downstream tasks. The seamless launch of a payable's registers, trial balance, and reconciliation reports represent some of the examples in practice as below.

- Automated payment data extract from Oracle ERP Cloud to update downstream external applications.
- Existing master data extracts, such as customers, suppliers, and so on, to synchronise with external applications.

Key Features

- Standard prebuilt reports across applications that can be generated on demand.

- BI Publisher report capabilities with custom reporting tools.
- Efficient data extract formats such as XML, CSV, and TEXT.
- Automation of end-to-end export flows with web service architecture.
- Tracking of export processes for completion, error tracking, and resubmission.
- Features that empower businesses with notification such as email, bell notification, business events and web service-based callback to initiate downstream business tasks or operations.

Details

Exporting data from Oracle ERP Cloud consists of the following steps:

1. Create a BI Publisher report(s) using the respective Enterprise Scheduler (ESS) job or BI Publisher dashboard.
2. Invoke the Oracle ERP Integration Service to initiate the respective export job.
3. Provide notification through asynchronous web service-based callback or business events upon completion.
4. Deliver the status and information using callback to access extracted data file(s) from the Oracle ERP Cloud. For alternate approach, see *Appendix 22: Export Financials Data without Using callback.*
5. Review any errors if applicable and take appropriate action, such as process the data extracted for further downstream business operation needs.

Automated End-to-End Outbound (Bulk Export) Orchestrated Flow

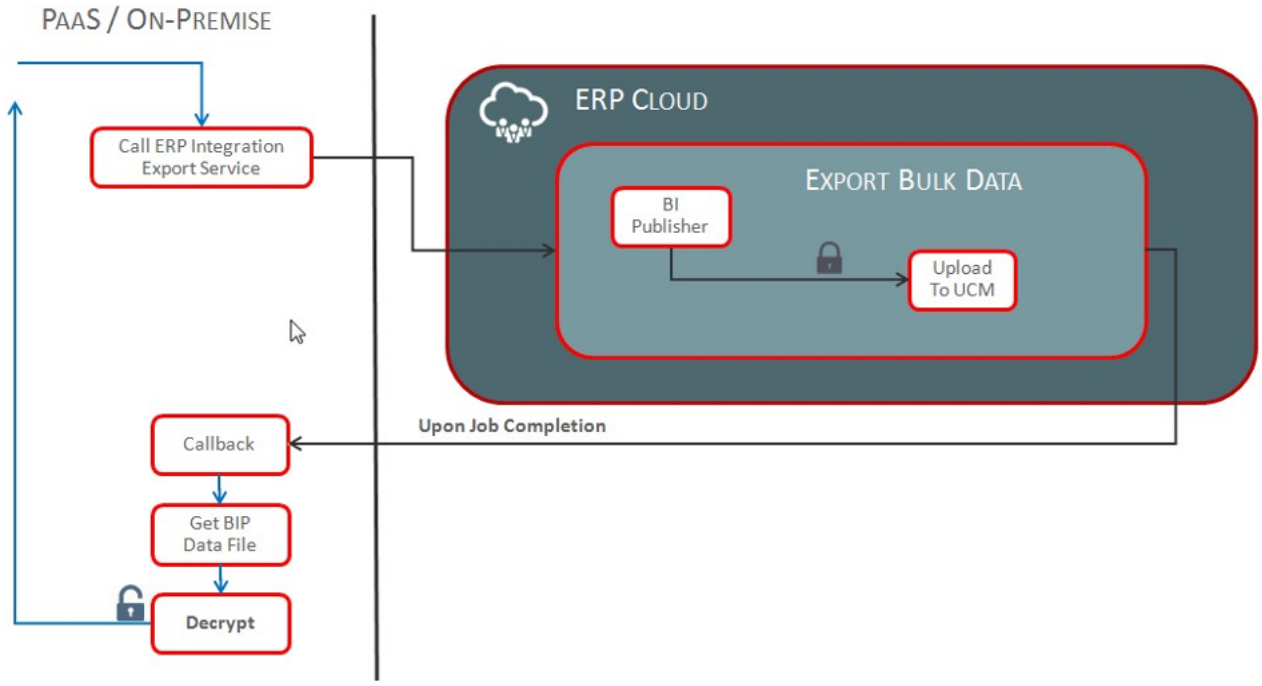


Figure 4: Outbound data integration orchestration flow

Example

To illustrate the outbound data integration with Oracle ERP Cloud, the "Receivable Billing History Extract" flow will be used as an example. A batch of transactions is extracted from the application and sent to the customers. The transactions are extracted from the output file of the Receivable Billing History Extract ESS process.

Prerequisites

- Import a certificate into your local keystore. For more information, see *Appendix 5: Testing Web Service using a client Proxy.*

- Configure the sample web service tester. For more information, see *Appendix 5: Testing Web Service using a client Proxy.*
- Identify the username and password to call the Receivable Billing History Extract.
- Verify that the user has access to the AttachmentsRead role in the UCM server. For more information, see *Appendix 1: Security Prerequisites to Download the Job Output File.*
- Verify the end point URL for the web service. For more information, see the automation web service at: *https://<hostname>.<domainname>/fscmService/ErpIntegrationService?WSDL.*

Flow Steps

1. Prepare the payload for the exportBulkData operation to request the data extract from Oracle ERP Cloud.
2. Receive a bell, email, or web-service based callback or business event notification for the request identifier returned by the web service operation in step 1.
3. Prepare the payload for the getDocumentForDocumentId operation to download the output file.

Flow Automation using the Oracle ERP Integration Web Service

A web service for Oracle Fusion Financials is an artifact that provides a standardized way of integrating other web-based applications or business system processes with Oracle ERP Cloud. Web services allow organizations to communicate with Oracle ERP Cloud without any application expertise. The Oracle ERP Integration Service is an external web service that provides robust web service operations, such as supporting the bulk import of data into the Oracle ERP Cloud, the bulk export of data from the Oracle ERP Cloud, and key value-added operations to retrieve files and purge interface and error data periodically.

To access automation details using the Oracle ERP Integration Service, refer to the SOAP Web Services guide for your cloud services on the Oracle Help Center (https://docs.oracle.com/pls/topic/lookup?ctx=fa-latest&id=OESWF).

Constructing the Oracle ERP Integration Service End Point URL

To obtain the physical end point of any specific instance:

1. Launch the ATK home page and sign in as a functional user.
2. Navigate to a dashboard or work area associated with the Payables Service.
3. In the Payables Invoice workbench, you can see a URL in the browser similar to https://<hostname>.<domainname>/payables/faces/InvoiceWorkbench.

   The "*<hostname>.<domainname>*" may be "*https://<pod-name>.<lba>.xxx.oraclecloud.com*".

   In this example "*<pod-name>.<lba>*" is the hostname and "*xxx.oraclecloud.com*" is the domain name.

   a. In this URL, capture "*https://<hostname>.<domainname>*".
   b. Append the static context root: "*/fscmService/ErpIntegrationService* ". "*https://<hostname>.<domainname>/fscmService/ErpIntegrationService*" is the WSDL URL for the Oracle ERP Integration Service.

Critical Web Service Operations to Automate Integration Flows

The Oracle ERP Integration Service provides multiple operations to aid inbound and outbound data integrations. The web service is available in both SOAP and REST. For more details, see these guides on the Oracle Help Center :

- SOAP ERP Integration Service Documentation: https://docs.oracle.com/pls/topic/lookup?ctx=fa-latest&id=OESWFREST ERP Integration
  Service Documentation: https://docs.oracle.com/pls/topic/lookup?ctx=fa-latest&id=FARFA

S.No	Operation Name	Summary
1	importBulkData	Initiates a process to upload bulk data in Oracle Fusion database
2	exportBulkData	Initiates a process to extract the selected data from the Oracle Fusion database
3	getDocumentForDocumentId	Extracts the document matching with provided document identifier from UCM
4	uploadFileToUCM	Uploads a document to the UCM server based on provided document details
5	submitESSJobRequest	Submits an ESS job request for the specified job definition
6	submitJobWithOutput	Submits an ESS job request for the specified job definition and further sends a web service-based call back to provided URL
7	getESSJobStatus	Extract the 'request status' of the provided ESS job
8	downloadESSJobExecutionDetails	Downloads the ESS job output and logs as a ZIP file for provided ESS request ID
9	getESSExecutionDetails	Obtains the ESS Job Execution details of the provided ESS Request ID.
10	getDocumentIdsForFilePrefix	Extracts the identifier of document/s having title starting with the provided file prefix
11	getDocumentsForFilePrefix	Extracts the documents details having title starting with the provided file prefix
12	appendFileComment	Appends the specified comment to a document corresponding to provided numeric document identifier

Operation: importBulkData

The importBulkData operation uploads a file to the Oracle Universal Content Management (UCM) server based on the document details specified and submits an ESS job to load and import the uploaded files to an application table.

The following table lists the parameters for this operation:

<table>
<tr><th>Parameter Name</th><th>Description</th><th>Parameter (In/Out)</th><th>Mandatory</th><th>Type</th></tr>
<tr><td>Document</td><td>List of elements, each containing the details of the file to be uploaded. The details include the file content, file name, content type, file title, author, security group, and account.

Mandatory Document Attributes:

• Content: File content uploaded on to the UCM server. The value of the content tag is obtained by converting the file content into Base64 encoding. For a sample program for Base64 encoding, see Appendix 2: Sample Code for Preparing a Data File for Inbound and Outbound Flow.
• FileName: Name of the file on the UCM server.

Optional Document Attributes:

• Content Type: Type of content uploaded such as zip, txt, dat, csv, xml, and ack.
• DocumentTitle: Title of the file on the UCM server.
• DocumentAuthor: Author of the document.
• DocumentSecurityGroup: A fixed value used to import or export documents on the UCM server. The security group for all the import processes is FAFusionImportExport.
• DocumentAccount: Account under which the file is uploaded.
• DocumentName: Name of the document. This is a unique field in UCM.
• DocumentId: Only if the file is already upload to UCM. When the DocumentId is provided above attributes are not needed.</td><td>IN</td><td>Yes</td><td>java.lang .String</td></tr>
<tr><td>Job Details</td><td>The details of the ESS job used to import and process the uploaded file. The details include the primary job information (job definition name, job package name), ParameterList, and JobRequestId.

To get the job package and definition name, see Appendix 18: How to get job definition & package name including parameters.

• Job package name, job definition should be separated by comma (,) .
• Supported parameters are only submit.argument1 to submit.argumentN. The order of the parameters is maintained per the list i.e. submit.argument1, submit.argument2, submit.argument3 ... submit.argumentN.
• Parameters should be passed as comma(,) separated string or as #NULL if a parameter is null.</td><td>IN</td><td>No, if the job property file is provided</td><td>java.lang .String</td></tr>
<tr><td>Notification

Code</td><td>A two-digit number that determines how and when a bell or email notification is passed for the status of the import job. See the table below for the notification code values. Pass #NULL if a bell or email notification is not needed.
<table>
<tr><th>Digit Position</th><th>Digit Value</th><th>Meaning</th></tr>
<tr><td rowspan="3">First digit</td><td>1</td><td>Email notification</td></tr>
<tr><td>2</td><td>Bell notification</td></tr>
<tr><td>3</td><td>Email and bell notification</td></tr>
</table></td><td>IN</td><td>Yes</td><td>java.lang .String</td></tr>
</table>

<table>
<tr><th>Parameter Name</th><th>Description</th><th>Parameter (In/Out)</th><th>Mandatory</th><th>Type</th></tr>
<tr><td></td><td>
<table>
<tr><td>Second digit</td><td>0</td><td>Send in any case (import failed or succeeded)</td></tr>
<tr><td></td><td>1</td><td>Send on import success</td></tr>
<tr><td></td><td>2</td><td>Send on import failure</td></tr>
</table>
</td><td></td><td></td><td></td></tr>
<tr><td>Callback URL</td><td>The web service-based callback URL of the web service you implemented to receive the ESS job status upon job completion. Pass #NULL if web service-based callback is not needed. See Appendix 10: Creating a Callback Web Service.</td><td>IN</td><td>No</td><td>java.lang .String</td></tr>
<tr><td>Job Options</td><td>Optional parameters, comma separated. It designates the specific inbound bulk data processing variation. See the list below for supported Job Options and their significance.
<table>
<tr><th>Job Option</th><th>Significance</th><th>Supported Values</th></tr>
<tr><td>InterfaceDetail s</td><td>InterfaceOptionId for which you want to run the importBulkData.
(Positive Numeric value)</td><td>See Appendi x 21: Supported Import Processes by Load Interface File for Import Job</td></tr>
<tr><td>ImportOption</td><td>Determines if provided import job(s) will be submitted or not.
Default value is Y.</td><td>Y, N</td></tr>
<tr><td>PurgeOption</td><td>Determines if Purge Interface Tables job will run post load.
Default value is N.</td><td>Y, N</td></tr>
<tr><td>FileEncryption
FA_ALIAS
CUSTOMER_ALIAS</td><td>To enable data file encryption, you must provide the following job options:
FileEncryption=PGPUNSIG NED or PGPSIGNED
FA_ALIAS=<ERP Cloud Key Alias Name></td><td>PGPUNSIGN ED, PGPSIGNED for FileEncrypti on.
FA_ALIAS, CUSTOMER _ALIAS are</td></tr>
</table>
</td><td>IN</td><td>No</td><td>java.lang .String</td></tr>
</table>

Parameter Name	Description			Parameter (In/Out)	Mandatory	Type
		CUSTOMER_ALIAS=<Customer Key Alias Name> Example: FileEncryption=PGPUNSIGNED,FA_ALIAS=ERP_CLOUD_KEY, CUSTOMER_ALIAS=CUSTOMER_ERP_KEY	custom as per customer created Keys.			
	EnableEvent	Determines if business event will be sent after completion of all ESS Jobs. Default value is N.	Y, N			
	EventIncludeImportJob	Determines if business event, web service-based payload will include First Import Job details even if not submitted. Default value is N.	Y, N			
	ExtractFileType	Determines if Upload Interface Error and Job Output File to Universal Content Management job will be submitted or not. *When NONE, the job is not submitted. *For other values the job downloads ESS OUT and/or LOG and/or ERROR files of load job and import jobs, compresses together and uploads to the corresponding UCM account. Default value is ALL if PurgeOption=Y, otherwise NONE.	ALL, OUT, LOG, NONE, ERROR			
	JobDetailFileName	Name of the manifest file uploaded to UCM that contains the import job details. See *Appendix 18: How to get job definition & package name*				

Parameter Name	Description	Parameter (In/Out)	Mandatory	Type
Response Code	The response code that returns the request identifier for the first job in the joblist, which is a Load Interface File for Import Job. **Note:** When a file upload to the UCM server fails, the remaining ESS jobs aren't executed, and an error is returned.	OUT		java.lang .Long

Job Details

The job details include the job definition and package names, as well as the job parameters of the imported object. The following options may be used to specify the Job Details parameter associated with the importBulkData operation:

- Specify the Job Details parameter directly in the request payload.
- Add the Job Property file as part of the data ZIP file.
- Upload the Job Properties file to the UCM and add JobDetailFileName=<FileName> in jobOptions

To get the job package, definition name, and list of parameters, See *Appendix 18: How to get job definition & package name including parameters.*

To generate a property file, see *Appendix 11: Creating a Job Property File for the importBulkData Operation.*

The following example illustrates the Journal Import process with the parameters included in the request payload:

ImportBulkData Request Payload

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns1:importBulkData
xmlns:ns1="http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationServic
e/types/">
      <ns1:document
xmlns:ns2="http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationServic
e/">
        <ns2:Content>BASE64 ENCODED STRING OF THE DATA FILE
JournalsImportTEST_1234.zip</ns2:Content>
        <ns2:FileName>JournalsImportTEST_1234.zip</ns2:FileName>
        <ns2:ContentType>zip</ns2:ContentType>
        <ns2:DocumentTitle>JournalsImportTEST_1234.zip</ns2:DocumentTitle>
        <ns2:DocumentAuthor>TestUser1</ns2:DocumentAuthor>
        <ns2:DocumentSecurityGroup>FAFusionImportExport</ns2:DocumentSecurityGroup>
        <ns2:DocumentAccount>fin$/generalLedger$/import$</ns2:DocumentAccount>
      </ns1:document>
      <ns1:jobDetails
xmlns:ns2="http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationServic
e/">
```

```
<ns2:JobName>/oracle/apps/ess/financials/generalLedger/programs/common,JournalImportLauncher</ns2:JobName>
        <ns2:ParameterList>GL_BU_ABC,1061,Balance Transfer,1,ALL,Y,Y,N</ns2:ParameterList>
        <ns2:JobRequestId/>
      </ns1:jobDetails>
      <ns1:notificationCode>30</ns1:notificationCode>
      <ns1:callbackURL>#NULL</ns1:callbackURL>
      <ns1:jobOptions>ImportOption= Y ,PurgeOption =  N ,
ExtractFileType=ALL,InterfaceDetails=15</ns1:jobOptions>
    </ns1:importBulkData>
  </soap:Body>
</soap:Envelope>
```

Figure 5: Sample request payload for the Journals Import process

The importBulkData operation response contains the Request ID of the job loading data into the respective product interface table.

ImportBulkData Response Payload

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsa="http://www.w3.org/2005/08/addressing">
  <env:Header>

<wsa:Action>http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationService/ErpIntegrationService/importBulkDataResponse</wsa:Action>
    <wsa:MessageID>urn:uuid:07a96e2e-09d1-44c5-82fe-08f3a504b260</wsa:MessageID>
  </env:Header>
  <env:Body>
    <ns0:importBulkDataResponse
xmlns:ns0="http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationService/types/">
      <result
xmlns="http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationService/types/">4084033</result>
    </ns0:importBulkDataResponse>
  </env:Body>
</env:Envelope>
```

Figure 6: Sample response payload for the importBulkData web service operation

Operation: exportBulkData

The exportBulkData operation submits an ESS job request for the specified job definition and parameters and returns the ESS RequestID. After completion of the provided ESS job, it also submits the "Upload Interface Error and Job Output File to Universal Content Management" job, which downloads ESS OUT and LOG files generated by the given ESS job, compresses together and uploads to UCM.

The following table lists the parameters for the exportBulkData operation:

<table>
<tr><th>Parameter Name</th><th>Description</th><th>Parameter (In/Out)</th><th>Mandatory</th><th>Type</th></tr>
<tr><td>Job Name</td><td>Job package name, Job definition name, both commas separated.

Example:
oracle/apps/ess/financials/commonModules/shared/common/outbound/,ReceivablesBillingHistoryExtract</td><td>IN</td><td>Yes</td><td>java.lang.String</td></tr>
<tr><td>Parameter List</td><td>ESS job parameters of the ESS job, comma separated. Supported parameters are only submit.argument1 to submit.argumentN. The order of the parameters is maintained per the list i.e., submit.argument1, submit.argument2, submit.argument3 ... submit.argumentN . Pass #NULL if a parameter is null. If the job does not have parameters, leave the tag empty.</td><td>IN</td><td>Yes</td><td>java.lang.String</td></tr>
<tr><td>Notification Code</td><td>A two-digit number that determines how and when a notification is passed for the status of the export job. See the table below for the notification code values. Pass #NULL if a bell or email notification is not needed.
<table>
<tr><th>Digit Position</th><th>Digit Value</th><th>Meaning</th></tr>
<tr><td rowspan="3">First digit</td><td>1</td><td>Email notification</td></tr>
<tr><td>2</td><td>Bell notification</td></tr>
<tr><td>3</td><td>Email and bell notification</td></tr>
<tr><td rowspan="3">Second digit</td><td>0</td><td>Send in any case (import failed or succeeded)</td></tr>
<tr><td>1</td><td>Send on import success</td></tr>
<tr><td>2</td><td>Send on import failure</td></tr>
</table></td><td>IN</td><td>No</td><td>java.lang.String</td></tr>
<tr><td>Callback URL</td><td>The web service-based callback URL of the web service you implemented to receive the ESS job status upon job completion. Pass #NULL if web service-based callback is not needed. See Appendix 10: Creating a Callback Web Service.</td><td>IN</td><td>No</td><td>java.lang.String</td></tr>
<tr><td>Job Options</td><td>Optional parameters, comma separated. See the list below for supported Job Options and their significance.</td><td>IN</td><td>No</td><td>java.lang.String</td></tr>
</table>

	<table><tr><th>Job Option</th><th>Significance</th><th>Supported Values</th></tr><tr><td>File Encryption FA_ALIAS CUSTOMER_ALIAS</td><td>To enable data file encryption, you must provide the following job options: FileEncryption=PGPUNSIGNED or PGPSIGNED FA_ALIAS=<ERP Cloud Key Alias Name> CUSTOMER_ALIAS=<Customer Key Alias Name> Example: FileEncryption=PGPUNSIGNED, FA_ALIAS=ERP_CLOUD_KEY, CUSTOMER_ALIAS=CUSTOMER_ERP_KEY</td><td>PGPUNSIGNED, PGPSIGNED for FileEncryption. FA_ALIAS, CUSTOMER_ALIAS are custom as per customer created Keys.</td></tr><tr><td>EnableEvent</td><td>Determines if business event will be sent after completion of all ESS Jobs. Default value N.</td><td>Y, N</td></tr><tr><td>ExtractFileType</td><td>Only Supported with DEV OPT IN .</td><td>Refer Appendix 19: ExportBulkData Operation Precise ExtractFileType Support</td></tr></table>			
Response Code	The response code that returns the request identifier of the export job.		OUT		java.lang.Long

The following illustration highlights a sample request payload of the exportBulkData operation to run "Receivable Billing History Extract" job:

ExportBulkData Response Payload

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:typ="http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationService/
types/">
  <soapenv:Header/>
  <soapenv:Body>
    <typ:exportBulkData>

<typ:jobName>oracle/apps/ess/financials/commonModules/shared/common/outbound/,ReceivablesBilling
HistoryExtract</typ:jobName>

<typ:parameterList>204,#NULL,#NULL,#NULL,#NULL,#NULL,#NULL,#NULL,#NULL,#NULL,#NULL,#NULL,
#NULL,1,FULL_EXTRACT,#NULL,#NULL,#NULL,ReceivablesBillingHistoryExtract,#NULL</typ:parameterList>

      <typ:jobOptions>EnableEvent=Y</typ:jobOptions>
      <typ:callbackURL>#NULL</typ:callbackURL>
      <typ:notificationCode>#NULL</typ:notificationCode>
    </typ:exportBulkData>
  </soapenv:Body>
</soapenv:Envelope>
```

Figure 7: Sample request payload for the exportBulkData operation

ExportBulkData Request Payload

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsa="http://www.w3.org/2005/08/addressing">
  <env:Header>

<wsa:Action>http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationServic
e/ErpIntegrationService/exportBulkDataResponse</wsa:Action>
    <wsa:MessageID>urn:uuid:2ba8bf18-9fa5-40c4-8cfd-5b64008a3d2d</wsa:MessageID>
  </env:Header>
```

```
<env:Body>
  <ns0:exportBulkDataResponse
xmlns:ns0="http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationService
/types/">
    <result
xmlns="http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationService/typ
es/">4084081</result>
  </ns0:exportBulkDataResponse>
</env:Body>
</env:Envelope>
```

Figure 8: Sample response payload for the exportBulkData operation

NOTE: To see advance options for ExportBulkData operation, see *Appendix 19: ExportBulkData Operation Precise ExtractFileType Support*.

Operation: getDocumentForDocumentId

The getDocumentForDocumentId operation returns the document that matches the provided document ID. This operation should be used downloads the job output file generated by the importBulkData operation or data file extracted by exportBulkData operation based on the documentId provided in the web service-based callback or business event.

This operation requires application user access and access to the AttachmentsRead role to download data file extracted by exportBulkData operation.

This operation requires application user access and the relevant role to access the corresponding UCM account to downloads the job output file generated by the importBulkData operation.

For more information on assigning a user with this access, see *Appendix 1: Security Prerequisites to Download the Job Output File.*

Note: Only files uploaded to Attachments or any UCM Account of FAFusionImportExport security groups can be downloaded through any ERP Integration Service.

The following table lists the parameters for this operation:

Parameter Name	Description	Parameter (In/Out)	Mandatory	Type
Document ID	The UCM document ID from the web-service based callback or business event response.	IN	Yes	java.lang.String
return	A list of elements, each containing the details of the downloaded files. The details include the document	OUT		List<DocumentDetailsVORowImpl>

Parameter Name	Description	Parameter (In/Out)	Mandatory	Type
	ID, file content, file name, content type, file title, author, security group, and account.			

The following sample request payload illustrates the getDocumentForDocumentId operation:

GetDocumentForDocumentId Request Payload

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:typ="http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationService/
types/">
  <soapenv:Header/>
  <soapenv:Body>
    <typ:getDocumentForDocumentId>
      <typ:DocumentId>4013403</typ:DocumentId>
    </typ:getDocumentForDocumentId>
  </soapenv:Body>
</soapenv:Envelope>
```

Figure 9: Sample request payload for the getDocumentForDocumentId operation

GetDocumentForDocumentId Response Payload

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsa="http://www.w3.org/2005/08/addressing">
  <env:Header>

<wsa:Action>http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationServic
e/ErpIntegrationService/getDocumentForDocumentIdResponse</wsa:Action>
    <wsa:MessageID>urn:uuid:330595ee-ad49-4591-9c7f-d0ab5346a149</wsa:MessageID>
  </env:Header>
  <env:Body>
    <ns0:getDocumentForDocumentIdResponse
xmlns:ns0="http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationService
/types/">
      <ns2:result xsi:type="ns0:DocumentDetails" xmlns:ns1="http://xmlns.oracle.com/adf/svc/types/"
xmlns:ns0="http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationService
/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
xmlns:ns2="http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationService
/types/">
        <ns0:Content>
          <xop:Include href="cid:3b79cc67-26b3-47c8-9631-cd0b3e0bf18c"
xmlns:xop="http://www.w3.org/2004/08/xop/include"/>
        </ns0:Content>
        <ns0:FileName xsi:nil="true"/>
        <ns0:ContentType xsi:nil="true"/>
        <ns0:DocumentTitle>ImportBulkData_JournalImportLauncher_4084033</ns0:DocumentTitle>
        <ns0:DocumentAuthor>suraj.singh@oracle.com</ns0:DocumentAuthor>
        <ns0:DocumentSecurityGroup>FAFusionImportExport</ns0:DocumentSecurityGroup>
        <ns0:DocumentAccount>fin$/generalLedger$/import$</ns0:DocumentAccount>
        <ns0:DocumentName>ImportBulkData_JournalImportLauncher_4084033.zip</ns0:DocumentName>
        <ns0:DocumentId>4013403</ns0:DocumentId>
      </ns2:result>
    </ns0:getDocumentForDocumentIdResponse>
  </env:Body>
</env:Envelope>
```

Figure 10: Sample response payload for the getDocumentForDocumentId operation

Operation: uploadFileToUCM

The uploadFileToUCM operation uploads a file to the UCM server based on the document specified. This operation requires application user access and the relevant role to access the corresponding UCM account under FAFusionImportExport security group.

Parameter Name	Description	Parameter (In/Out)	Is Mandatory	Type
Document	List of elements, each containing the details of the file to be uploaded. The details include the file content, file name, content type, file title, author, security group, and account. Mandatory Document Attributes: • Content: File content uploaded on to the UCM server. The value of the content tag is obtained by converting the file content into Base64 encoding.	IN	Yes	java.lang.String

Parameter Name	Description	Parameter (In/Out)	Is Mandatory	Type
	• FileName: Name of the file on the UCM server. Optional Document Attributes: • ContentType: Type of content uploaded such as zip, txt, or csv. • DocumentTitle: Title of the file on the UCM server. • DocumentAuthor: Author of the document. • DocumentSecurityGroup: A fixed value used to import or export documents on the UCM server. The security group for all the import processes is FAFusionImportExport. • DocumentAccount: Account under which the file is uploaded. • DocumentName: Name of the document. This is a unique field in UCM.			
Return	Returns the document ID of the uploaded file.	OUT	No	java.lang.String

Note: Files can be uploaded to any UCM Account of the FAFusionImportExport security group only.

UploadFileToUcm Request Payload

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns1:uploadFileToUcm
xmlns:ns1="http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationService
/types/">
      <ns1:document
xmlns:ns2="http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationService
/">


                        <ns2:Content>BASE64 ENCODED STRING OF THE DATA FILE
JournalsImportTEST_1234.zip</ns2:Content>
                    <ns2:FileName>JournalsImportTEST_1234.zip</ns2:FileName>
        <ns2:ContentType>zip</ns2:ContentType>
        <ns2:DocumentTitle>JournalsImportTEST_1234.zip</ns2:DocumentTitle>
```

```
        <ns2:DocumentAuthor>TestUser</ns2:DocumentAuthor>
        <ns2:DocumentSecurityGroup>FAFusionImportExport</ns2:DocumentSecurityGroup>
        <ns2:DocumentAccount>fin$/generalLedger$/import$</ns2:DocumentAccount>
      </ns1:document>
    </ns1:uploadFileToUcm>
  </soap:Body>
</soap:Envelope>
```

Figure 11: Sample request payload for the UploadFileToUcm operation

UploadFileToUcm Response Payload

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsa="http://www.w3.org/2005/08/addressing">
  <env:Header>

<wsa:Action>http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationServic
e/ErpIntegrationService/uploadFileToUcmResponse</wsa:Action>
    <wsa:MessageID>urn:uuid:0dbff259-eff6-47a4-9ff0-f012ef7b640d</wsa:MessageID>
  </env:Header>
  <env:Body>
    <ns0:uploadFileToUcmResponse
xmlns:ns0="http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationService
/types/">
      <result
xmlns="http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationService/typ
es/">4013427</result>
    </ns0:uploadFileToUcmResponse>
  </env:Body>
</env:Envelope>
```

Figure 12: Sample response payload for the UploadFileToUcm operation

Operation: submitESSJobRequest

The submitESSJobRequest operation submits an ESS job request for the specified job definition. This operation requires the user to have access on the specified ESS job.

Parameter Name	Description	Parameter (In/Out)	Is Mandatory	Type
jobPackageName	The ESS job package name for the ESS job.	IN	Yes	java.lang.String
jobDefinitionName	The ESS job definition name for the ESS job.	IN	Yes	java.lang.String
paramList	List of parameters used to invoke the ESS job. Supported parameters are only submit.argument1 to submit.argumentN. The order of the parameters is maintained per the list i.e. submit.argument1, submit.argument2, submit.argument3 ... submit.argumentN . The corresponding entry in the list should be blank or passed as #NULL when a given parameter need not be passed or is null.	IN	No	java.lang.String
Return	The response code that returns the request identifier of the ESS job.	OUT	Yes	java.lang.String

SubmitESSJobRequest Request Payload

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body>
        <ns1:submitESSJobRequest
xmlns:ns1="http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationService
/types/">

<ns1:jobPackageName>oracle/apps/ess/financials/commonModules/shared/common/interfaceLoader</ns
1:jobPackageName>
            <ns1:jobDefinitionName>InterfaceLoaderController</ns1:jobDefinitionName>
            <ns1:paramList>15</ns1:paramList>
            <ns1:paramList>4013405</ns1:paramList>
            <ns1:paramList>N</ns1:paramList>
            <ns1:paramList>N</ns1:paramList>
    </ns1:submitESSJobRequest>
  </soap:Body>
</soap:Envelope>
```

Figure 13: Sample request payload for the SubmitESSJobRequest operation

SubmitESSJobRequest Response Payload

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsa="http://www.w3.org/2005/08/addressing">
  <env:Header>

<wsa:Action>http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationServic
e/ErpIntegrationService/submitESSJobRequestResponse</wsa:Action>
    <wsa:MessageID>urn:uuid:29176c05-d323-4ba7-8695-8c7502930a88</wsa:MessageID>
  </env:Header>
  <env:Body>
    <ns0:submitESSJobRequestResponse
xmlns:ns0="http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationService
/types/">
      <result
xmlns="http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationService/typ
es/">4084062</result>
    </ns0:submitESSJobRequestResponse>
  </env:Body>
</env:Envelope>
```

Figure 14: Sample response payload for the SubmitESSJobRequest operation

Operation: submitJobWithOutput

The submitJobWithOutput operation submits an ESS job request for the specified job definition and sends a web-service based callback to the specified callback URL when the job completes. This operation requires the user to have access on the specified ESS job.

Parameter Name	Description	Parameter (In/Out)	Is Mandatory	Type
jobPackageName	The ESS job package name for the ESS job.	IN	Yes	java.lang.String
jobDefinitionName	The ESS job definition name for the ESS job.	IN	Yes	java.lang.String
callbackURL	The web service based callback URL of the web service you implemented to receive the ESS job status upon job completion. Pass #NULL if web	IN	No	java.lang.String

Parameter Name	Description	Parameter (In/Out)	Is Mandatory	Type
	service based callback is not needed. See *Appendix 10: Creating a Callback Web Service.*			
paramList	List of parameters used to invoke the ESS job. Supported parameters are only submit.argument1 to submit.argumentN. The order of the parameters is maintained per the list i.e., submit.argument1, submit.argument2, submit.argument3 ... submit.argumentN . The corresponding entry in the list should be blank or passed as #NULL when a given parameter need not be passed or is null.	IN	No	java.lang.String
Return	The response code that returns the request identifier of the ESS job.	OUT	Yes	java.lang.String

SubmitJobWithOutput Request Payload

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body>
        <ns1:submitJobWithOutput
xmlns:ns1="http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationService
/types/">

<ns1:jobPackageName>/oracle/apps/ess/financials/generalLedger/programs/common</ns1:jobPackageNa
me>
            <ns1:jobDefinitionName>JournalImportLauncher</ns1:jobDefinitionName>
            <ns1:callbackURL>PASS CALLBACK URL HERE</ns1:callbackURL>
                                        <ns1:paramList>GL_BU_ABC</ns1:paramList>
            <ns1:paramList>1061</ns1:paramList>
            <ns1:paramList>Balance Transfer</ns1:paramList>
            <ns1:paramList>1</ns1:paramList>
            <ns1:paramList>ALL</ns1:paramList>
            <ns1:paramList>Y</ns1:paramList>
            <ns1:paramList>Y</ns1:paramList>
            <ns1:paramList>N</ns1:paramList>
```

```
    </ns1:submitJobWithOutput>
  </soap:Body>
</soap:Envelope>
```

Figure 15: Sample request payload for the SubmitJobWithOutput operation

SubmitJobWithOutput Response Payload

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsa="http://www.w3.org/2005/08/addressing">
  <env:Header>

<wsa:Action>http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationServic
e/ErpIntegrationService/submitJobWithOutputResponse</wsa:Action>
    <wsa:MessageID>urn:uuid:39acab17-dca9-4208-ad41-2a8add461ace</wsa:MessageID>
  </env:Header>
  <env:Body>
    <ns0:submitJobWithOutputResponse
xmlns:ns0="http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationService
/types/">
      <result
xmlns="http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationService/typ
es/">4084067</result>
    </ns0:submitJobWithOutputResponse>
  </env:Body>
</env:Envelope>
```

Figure 16: Sample response payload for the SubmitJobWithOutput operation

Operation: getESSJobStatus

The getEssJobStatus method obtains the execution status for a specified ESS request ID. This operation requires the user to have access on the specified ESS job.

Parameter Name	Description	Parameter (In/Out)	Is Mandatory	Type
requestID	The request ID of the ESS job.	IN	Yes	java.lang.String

Parameter Name	Description	Parameter (In/Out)	Is Mandatory	Type
Return	Returns the current status of the ESS job.	OUT	Yes	java.lang.String

GetESSJobStatus Request Payload

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:typ="http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationService/
types/">
  <soapenv:Header/>
  <soapenv:Body>
    <typ:getESSJobStatus>
      <typ:requestId>4084067</typ:requestId>
    </typ:getESSJobStatus>
  </soapenv:Body>
</soapenv:Envelope>
```

Figure 17: Sample request payload for the GetESSJobStatus operation

GetESSJobStatus Response Payload

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsa="http://www.w3.org/2005/08/addressing">
  <env:Header>

<wsa:Action>http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationServic
e/ErpIntegrationService/getESSJobStatusResponse</wsa:Action>
    <wsa:MessageID>urn:uuid:e8cf2ae4-abc7-44c6-b01d-3ae57eda8eba</wsa:MessageID>
  </env:Header>
  <env:Body>
    <ns0:getESSJobStatusResponse
xmlns:ns0="http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationService
/types/">
      <result
xmlns="http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationService/typ
es/">SUCCEEDED</result>
    </ns0:getESSJobStatusResponse>
  </env:Body>
```

```
</env:Envelope>
```

Figure 18: Sample response payload for the GetESSJobStatus operation

Operation: downloadESSJobExecutionDetails

The downloadESSJobExecutionDetails operation downloads the ESS job output and logs as a ZIP file for a specified ESS request ID. This operation requires the user to have the “AttachmentsRead” role.

Note: For Outbound Integration to export data from Oracle ERP (typical involving BIP job), it is recommended to use export bulk data as described in Operation: exportBulkData

Parameter Name	Description	Parameter (In/Out)	Is Mandatory	Type
requestID	The request ID of the ESS job.	IN	Yes	java.lang.String
fileType	The file type used to determine the execution details to download. Supported file types are LOG, OUT, ALL to download only LOG, only OUT, both LOG and OUT files respectively. Default value is ALL.	IN	No	java.lang.String
Return	Returns a ZIP file having the ESS job output and the logs of the specified ESS Job request.	OUT	Yes	java.lang.String

DownloadESSJobExecutionDetails Request Payload

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:typ="http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationService/
types/">
  <soapenv:Header/>
  <soapenv:Body>
    <typ:downloadESSJobExecutionDetails>
      <typ:requestId>4084062</typ:requestId>
      <typ:fileType>ALL</typ:fileType>
    </typ:downloadESSJobExecutionDetails>
  </soapenv:Body>
</soapenv:Envelope>
```

Figure 19: Sample request payload for the DownloadESSJobExecutionDetails operation

DownloadESSJobExecutionDetails Response Payload

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsa="http://www.w3.org/2005/08/addressing">
  <env:Header>

<wsa:Action>http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationServic
e/ErpIntegrationService/downloadESSJobExecutionDetailsResponse</wsa:Action>
    <wsa:MessageID>urn:uuid:14455965-15a3-4872-85e3-d401519eac68</wsa:MessageID>
  </env:Header>
  <env:Body>
    <ns0:downloadESSJobExecutionDetailsResponse
xmlns:ns0="http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationService
/types/">
      <ns2:result xsi:type="ns0:DocumentDetails" xmlns:ns1="http://xmlns.oracle.com/adf/svc/types/"
xmlns:ns0="http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationService
/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ns2="http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationService
/types/">
        <ns0:Content>
          BASE 64 ENCODED STRING OF 82105.zip file
        </ns0:Content>
                          <ns0:FileName xsi:nil="true"/>
        <ns0:ContentType>zip</ns0:ContentType>
        <ns0:DocumentTitle>ESS_L_4084062</ns0:DocumentTitle>
        <ns0:DocumentAuthor>suraj.singh@oracle.com</ns0:DocumentAuthor>
        <ns0:DocumentSecurityGroup>Attachments</ns0:DocumentSecurityGroup>
        <ns0:DocumentAccount xsi:nil="true"/>
        <ns0:DocumentName>4084062.zip</ns0:DocumentName>
        <ns0:DocumentId xsi:nil="true"/>
                  </ns2:result>
    </ns0:downloadESSJobExecutionDetailsResponse>
  </env:Body>
</env:Envelope>
```

Figure 20: Sample response payload for the DownloadESSJobExecutionDetails operation

Operation: getESSExecutionDetails

The getESSExecutionDetails method obtains Job Path, Name, Status, and all related child job details for a specified request ID. This operation requires the user to have access on the specified ESS job.

Parameter Name	Description	Parameter (In/Out)	Is Mandatory	Type
requestID	The request ID of the ESS job.	IN	Yes	java.lang.String
jobOptions	Optional parameters, comma separated. See the list below for supported Job Options and their significance.	IN	No	java.lang.String
Return	Returns the Job Path, Name, Status and all related child job details for a specified request ID.	OUT	Yes	java.lang.String

GetESSExecutionDetails Request Payload

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:typ="http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationService/
types/">
  <soapenv:Header/>
  <soapenv:Body>
    <typ:getESSExecutionDetails>
      <typ:requestId>4084062</typ:requestId>
      <typ:jobOptions></typ:jobOptions>
    </typ:getESSExecutionDetails>
  </soapenv:Body>
</soapenv:Envelope>
```

Figure 21: Sample request payload for the GetESSExecutionDetails operation

GetESSExecutionDetails Response Payload

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsa="http://www.w3.org/2005/08/addressing">
  <env:Header>

<wsa:Action>http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationServic
e/ErpIntegrationService/getESSExecutionDetailsResponse</wsa:Action>
    <wsa:MessageID>urn:uuid:2e1dcbdf-991d-45eb-a51e-4bc2de8d225f</wsa:MessageID>
```

```
  </env:Header>

  <env:Body>

    <ns0:getESSExecutionDetailsResponse
xmlns:ns0="http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationService
/types/">

      <result
xmlns="http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationService/typ
es/">{"JOBS":{"JOBNAME":"Load Interface File for
Import","JOBPATH":"/oracle/apps/ess/financials/commonModules/shared/common/interfaceLoader","REQ
UESTID":"4084062","STATUS":"SUCCEEDED","CHILD":[{"JOBNAME":"Load File to
Interface","JOBPATH":"/oracle/apps/ess/financials/commonModules/shared/common/interfaceLoader","RE
QUESTID":"4084064","STATUS":"SUCCEEDED"},{"JOBNAME":"Transfer
File","JOBPATH":"/oracle/apps/ess/financials/commonModules/shared/common/interfaceLoader","REQUES
TID":"4084063","STATUS":"SUCCEEDED"}]}}</result>

    </ns0:getESSExecutionDetailsResponse>

  </env:Body>

</env:Envelope>
```

Figure 22: Sample response payload for the GetESSExecutionDetails operation

Operation: getDocumentIdsForFilePrefix

The getDocumentIdsForFilePrefix operation returns the document identifiers of the files having document title starting with the specific file prefix from the provided UCM account under FAFusionImportExport security group not having the given comment. If the UCM account is specified as #NULL then operation gets the document identifiers of the files having document title starting with the specific file prefix from Attachment security group not having the given comment.

This operation can be used get documentID of the job output file generated by the importBulkData operation or data file extracted by exportBulkData operation based on file prefix ImportBulkData_<JobName>_<RequestId> or ExportBulkData_<JobName>_<RequestId>.

This operation requires application user access and access to the AttachmentsRead role to get the documentID of data file extracted by exportBulkData operation.

This operation requires application user access and the relevant role to access the corresponding UCM account to get the documentID of the job output file generated by the importBulkData operation.

For more information on assigning a user with this access, see *Appendix 1: Security Prerequisites to Download the Job Output File*.

Note: Only files uploaded to Attachments or any UCM Account of FAFusionImportExport security groups can be accessed through any ERP Integration Service.

Caution: A blank search should not be performed through this operation as it could lead to performance issues and out of memory issue. Pass values file prefix, UCM account, and Comment as precise as possible for best performance.

The following table lists the parameters for this operation:

Parameter Name	Description	Parameter (In/Out)	Mandatory	Type
prefix	The prefix with which the document title starts.	IN	Yes	java.lang.String
account	UCM account of the document if document belongs to the FAFusionImportExport security group. If the document belongs to the Attachment security group, pass #NULL	IN	No	java.lang.String
comments	The parameter to pass the comment. Files that do not have the comments are retrieved.	IN	No	java.lang.String
return	Returns a list of document identifiers.	OUT	Yes	List<java.lang.String>

The following sample request payload illustrates the getDocumentIdsForFilePrefix operation:

GetDocumentIdsForFilePrefix Request Payload

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:typ="http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationService/
types/">
  <soapenv:Header/>
  <soapenv:Body>
    <typ:getDocumentIdsForFilePrefix>
      <typ:prefix>ExportBulkData_ReceivablesBillingHistoryExtract_4084081</typ:prefix>
      <typ:account>#NULL</typ:account>
      <typ:comments>#NULL</typ:comments>
    </typ:getDocumentIdsForFilePrefix>
  </soapenv:Body>
</soapenv:Envelope>
```

Figure 23: Sample request payload for the getDocumentIdsForFilePrefix operation

GetDocumentIdsForFilePrefix Response Payload

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsa="http://www.w3.org/2005/08/addressing">
  <env:Header>

<wsa:Action>http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationServic
e/ErpIntegrationService/getDocumentIdsForFilePrefixResponse</wsa:Action>
    <wsa:MessageID>urn:uuid:8776cf8f-28ae-41b3-8d92-492644a11edf</wsa:MessageID>
  </env:Header>
  <env:Body>
    <ns0:getDocumentIdsForFilePrefixResponse
xmlns:ns0="http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationService
/types/">
      <result
xmlns="http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationService/typ
es/">4013456</result>
    </ns0:getDocumentIdsForFilePrefixResponse>
  </env:Body>
</env:Envelope>
```

Figure 24: Sample response payload for the getDocumentIdsForFilePrefix operation

Operation: getDocumentsForFilePrefix

The getDocumentsForFilePrefix operation returns the files having document title starting with the specific file prefix from the specified UCM account under the FAFusionImportExport security group doesn't have the value you provided for comment. If the UCM account is specified as #NULL, then the operation gets the files having document title starting with the specific file prefix from Attachment security group doesn't have the value you provided for comment.

This operation can be used downloads the job output file generated by the importBulkData operation or data file extracted by exportBulkData operation based on file prefix ImportBulkData_<JobName>_<RequestId> or ExportBulkData_<JobName>_<RequestId>.

This operation requires application user access and access to the AttachmentsRead role to download data file extracted by exportBulkData operation.

This operation requires application user access and the relevant role to access the corresponding UCM account to downloads the job output file generated by the importBulkData operation.

For more information on assigning a user with this access, see *Appendix 1: Security Prerequisites to Download the Job Output File.*

Note: Only files uploaded to Attachments or any UCM Account of FAFusionImportExport security groups can be downloaded through any ERP Integration Service.

Caution: A blank search should not be performed through this operation as it could lead to performance issues and out of memory issue. Pass values file prefix, UCM account, and comment as precise as possible for best performance.

The following table lists the parameters for this operation:

Parameter Name	Description	Parameter (In/Out)	Mandatory	Type
prefix	The prefix with which the document title starts.	IN	Yes	java.lang.String
account	UCM account of the document if document belongs to the FAFusionImportExport security group. In case document belongs to the Attachment security group, pass #NULL.	IN	No	java.lang.String
comments	The parameter to pass the comment. Files that do not have the comments are retrieved.	IN	No	java.lang.String
return	A list of elements, each containing the details of the downloaded files. The details include the document ID, file content, file name, content type, file title, author, security group, and account.	OUT	Yes	List<DocumentDetailsVORowImpl>

The following sample request payload illustrates the getDocumentsForFilePrefix operation:

GetDocumentsForFilePrefix Request Payload

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:typ="http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationService/
types/">
  <soapenv:Header/>
  <soapenv:Body>
    <typ:getDocumentsForFilePrefix>
      <typ:prefix>ImportBulkData_JournalImportLauncher_4084033</typ:prefix>
      <typ:account>fin$/generalLedger$/import$</typ:account>
```

```
      <typ:comments>#NULL</typ:comments>
    </typ:getDocumentsForFilePrefix>
  </soapenv:Body>
</soapenv:Envelope>
```

Figure 25: Sample request payload for the GetDocumentsForFilePrefix operation

GetDocumentsForFilePrefix Response Payload

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsa="http://www.w3.org/2005/08/addressing">
  <env:Header>

<wsa:Action>http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationServic
e/ErpIntegrationService/getDocumentsForFilePrefixResponse</wsa:Action>
    <wsa:MessageID>urn:uuid:ee988c8a-9e41-417e-b06e-ab2fa8fa0415</wsa:MessageID>
  </env:Header>
  <env:Body>
    <ns0:getDocumentsForFilePrefixResponse
xmlns:ns0="http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationService
/types/">
      <ns2:result xsi:type="ns0:DocumentDetails" xmlns:ns1="http://xmlns.oracle.com/adf/svc/types/"
xmlns:ns0="http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationService
/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ns2="http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationService
/types/">
        <ns0:Content>
          <xop:Include href="cid:7818efa8-24ee-4e2c-82e4-e4d2ad086601"
xmlns:xop="http://www.w3.org/2004/08/xop/include"/>
        </ns0:Content>
        <ns0:FileName xsi:nil="true"/>
        <ns0:ContentType xsi:nil="true"/>
        <ns0:DocumentTitle>ImportBulkData_JournalImportLauncher_4084033</ns0:DocumentTitle>
        <ns0:DocumentAuthor>suraj.singh@oracle.com</ns0:DocumentAuthor>
        <ns0:DocumentSecurityGroup>FAFusionImportExport</ns0:DocumentSecurityGroup>
        <ns0:DocumentAccount>fin$/generalLedger$/import$</ns0:DocumentAccount>
        <ns0:DocumentName>ImportBulkData_JournalImportLauncher_4084033.zip</ns0:DocumentName>
        <ns0:DocumentId>4013403</ns0:DocumentId>
      </ns2:result>
```

```
    </ns0:getDocumentsForFilePrefixResponse>
  </env:Body>
</env:Envelope>
```

Figure 26: Sample response payload for the GetDocumentsForFilePrefix operation

Operation: appendFileComment

Appends the specified comment to a document corresponding to provided numeric document identifier.

The following table lists the parameters for this operation:

Parameter Name	Description	Parameter (In/Out)	Mandatory	Type
documentIds	The numeric document identifier of a document.	IN	No	java.lang.String
comments	The comment that needs to be appended.	IN	No	java.lang.String

The following sample request payload illustrates the appendFileComment operation:

appendFileComment-request

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
        <soap:Body>
                <ns1:appendFileComment
xmlns:ns1="http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationService
/types/">
                        <ns1:documentId>4013427</ns1:documentId>

        <ns1:comments>FileEncryption=PGPUNSIGNED;FA_ALIAS=FUSION_AHC_IMPORT1;CUSTOMER_ALIA
S=FUSION_AHC_CUSTOMER1</ns1:comments>
    </ns1:appendFileComment>
  </soap:Body>
</soap:Envelope>
```

Figure 27: Sample request payload for the appendFileComment operation

ORACLE

Security Policies, Roles, Privileges

This section illustrates security policies, roles, privileges details required to use various artifacts involved during inbound and outbound integration design such as ERP Integration Service(SOAP/REST), ERP ICS Integration Service, ERP Processes Service, Load Interface File for Import ESS Job, Purge Interface Tables job etc.

Roles and Privileges

Category	Artifact	Security Tree (Privileges >> Role)
REST Services	erpintegrations	FUN_FSCM_REST_SERVICE_ACCESS_INTEGRATION_PRIV **>>** ORA_FUN_FSCM_LOAD_INTERFACE_ADMIN_DUTY
	erpprocesses	
SOAP Services	ErpIntegrationService	ALL_INTEGRATION_POINTS_ALL_DATA, authenticated-role
	ErpIcsIntegrationService	
ESS Jobs	Load Interface File for Import	FUN_FSCM_LOAD_INTERFACES_PRIV >> ORA_FUN_FSCM_LOAD_INTERFACE_ADMIN_DUTY
	Load Multiple Interface File for Import	
	Purge Interface Tables Job	
	Generate Data File for Export	FUN_FSCM_LOAD_INTERFACES_PRIV >> ORA_FUN_FSCM_LOAD_INTERFACE_ADMIN_DUTY ZX_IMPORT_TAXABLE_TRANSACTION_PRIV >> ORA_ZX_TAX_EXTERNAL_TRANSACTION_DUTY
	Payments Extract	
	Receivables Billing History Extract	
	Payables Transactions Extract	
	Receivables Transactions Extract	
	Receivables Adjustments Extract	
	Financial Tax Extract	
	Journals Extract	
	Receipts Analysis Extract	
	Trial Balance Extract	
	Asset Transfers Extract	
	Asset Additions Extract	

Category	Artifact	Security Tree (Privileges >> Role)
	Asset Retirements Extract	
	Asset Transactions History Extract	
	Bank Statement Extract	

Security Policies For ERP Integration SOAP Service and ERP ICS Integration SOAP Service

The Oracle ERP Integration Service is secured using the following policy: oracle/wss11_saml_or_username_token_with_message_protection_service_policy .

When a client calls the service, the service must satisfy the message protection policy to ensure that the payload is transported with encryption or sent over the SSL transport layer.

A client policy that can be used to meet this requirement is: oracle/wss11_username_token_with_message_protection_client_policy

To use this policy, the message must be encrypted using a public key provided by the server. When the message reaches the server, it can be decrypted by the server's private key. A keystore is used to import the certificate, and it is referenced in the subsequent client code.

The public key can be obtained from the certificate provided in the service WSDL file. See the following figure for an example of a certificate that is Base64 encoded.

```
<wsdl:service name="ErpIntegrationService">
<wsdl:port name="ErpIntegrationServiceSoapHttpPort" binding="tns:ErpIntegrationServiceSoapHttp">
<soap:address location="
[redacted]/publicFinancialCommonErpIntegration/ErpIntegrationService"/>
<wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing">
<wsa:Address xmlns:wsa="http://www.w3.org/2005/08/addressing">
[redacted]publicFinancialCommonErpIntegration/ErpIntegrationService
</wsa:Address>
<wsid:Identity xmlns:wsid="http://schemas.xmlsoap.org/ws/2006/02/addressingidentity">
<dsig:KeyInfo xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
<dsig:X509Data>
<dsig:X509Certificate>
MIIC6DCCAdCgAwIBAgIGAVP2HjwHMA0GCSqGSIb3DQEBCwUAMHgxCzAJBgNVBAYTAlVTMRAwDgYDVQQIEwdNeVN0YXR1MQ8wDQYDVQQHEwZNeVRvd24xFzA
VBgNVBAoTDk15T3JnYW5pemF0aW9uMRkwFwYDVQQLExBGT1IgVEVTVElORyBPTkxZMRIwEAYDVQQDEwlDZXJ0R2VuQ0EwHhcNMTYwNDA4MTM0NDQ5WhcNMj
EwNDA3MTM0NDQ5WjBTMRMwEQYKCZImiZPyLGQBGRYDY29tMRYwFAYKCZImiZPyLGQBGRYGb3JhY2xlMRIwEAYKCZImiZPyLGQBGRYCdXMxEDAOBgNVBAMTB
3NlcnZpY2UwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBANaq50W6Og6lHiTkhY2aJM/EX/BnbQHHt7RAkrQGtTJbK+Jvp2Un91Nf4j84nZyuE3gaxYpn
I4DM5jSOAw7oKCddvHs3XItgszmeCRyZLQGi9bQQuV1t8rK+7zn4Y4IsnNgzJrWmJ7fx1ATVYR7vrSgyQc1k1orK+15wAj9AS1jPAgMBAAGjITAfMB0GA1U
dDgQWBBRH4ZKiqNY5aBqqzhAXg1gUqJbHyzANBgkqhkiG9w0BAQsFAAOCAQEAHnvQAIDjSW95xa+J/vJmQUoK8W55dQt1ifPiQTh9AHoMTiouVug3k4gpXg
556BvrPKmni+YY+cN33uvsuXkypTh0mQydmlldiu5CZh2TPiDPFcpAOfxgw56sLT5q9PTJZJPwX952E7llTksILTBMECFmF/ChKpoTlyEqjrO+F+0CpsqvV
wIubPgF5qB06EBayGNCeVpoSX68RwOlJs80go2QoNJ0DV6U7vu7C9h/IkuRrQxCFhUSGOZr348CkX7sLtcBhrlIB5hJEb6F5d98kBQxRxcPvNWLE4ClMx5H
nMJnXnIq8oZrigTJNUlRo4iLfL4uYYhiQGZrIZukuQluyQ==
</dsig:X509Certificate>
<dsig:X509IssuerSerial>
<dsig:X509IssuerName>
CN=CertGenCA, OU=FOR TESTING ONLY, O=MyOrganization, L=MyTown, ST=MyState, C=US
</dsig:X509IssuerName>
<dsig:X509SerialNumber>1460123089927</dsig:X509SerialNumber>
</dsig:X509IssuerSerial>
<dsig:X509SubjectName>CN=service, DC=us, DC=oracle, DC=com</dsig:X509SubjectName>
<dsig:X509SKI>R+GSoqjWOWgaqs4QF4NYFKiWx8s=</dsig:X509SKI>
</dsig:X509Data>
</dsig:KeyInfo>
</wsid:Identity>
</wsa:EndpointReference>
</wsdl:port>
```

Figure 28: Example of a certificate in the Oracle ERP Integration Service WSDL file

To use the key contained in this certificate, create a local Keystore and import the certificate into it. For more information, see *Appendix 5: Testing Web Service using a Client Proxy.*

Note: The ERP ICS Integration SOAP Service follows the same security policy as ERP Integration SOAP Service. Importing the certificates for any one of the services is enough.

Callback Web Service

In practice, customers create and host a callback web service to optimally leverage the callback capabilities provided by the Oracle ERP Integration Service for notification purposes. The customer callback web service must implement the onJobCompletion() operation. When a job completes, Oracle ERP Integration Service invokes the customer callback web service as defined in the request payload of supported operations with callback capabilities, such as the importBulkData operation.

For more information on Callback Service, see *Appendix 10: Creating a Callback Web Service.*

Callback Response in JSON Format

The callback response provides execution statuses and request IDs of all the applicable ESS jobs. For example, to import AP invoices, the following jobs are executed:

1. Load Interface File for Import
    a. Transfer File
    b. Load Data to Interface tables
2. Import Invoices
    a. Import Payable Invoices
3. Upload Interface Error and Job Output File to Universal Content Management

The response includes the UCM document ID of the ZIP file uploaded by the "Upload Interface Error and Job Output File to Universal Content Management" job which contains output artifacts, such as the LOGs and OUT files of each ESS jobs, and data from the interface and error tables. This ZIP file can be retrieved using the Operation: getDocumentForDocumentId.

The following callback response provides the Request ID and status of each of the jobs outlined above.

Sample Callback Payload

```
{
  "JOBS": [
    {
      "JOBNAME": "Load Interface File for Import",
      "JOBPATH": "/oracle/apps/ess/financials/commonModules/shared/common/interfaceLoader",
      "DOCUMENTNAME": "APTEST_0310.zip",
      "REQUESTID": "455622",
      "STATUS": "SUCCEEDED",
      "CHILD": [
        {
          "JOBNAME": "Transfer File",
          "JOBPATH": "/oracle/apps/ess/financials/commonModules/shared/common/interfaceLoader",
          "REQUESTID": "455623",
          "STATUS": "SUCCEEDED"
        },
        {
          "JOBNAME": "Load File to Interface",
          "JOBPATH": "/oracle/apps/ess/financials/commonModules/shared/common/interfaceLoader",
          "REQUESTID": "455624",
          "STATUS": "SUCCEEDED"
        }
      ]
    },
    {
      "JOBNAME": "Import Payables Invoices",
```

```
      "JOBPATH": "/oracle/apps/ess/financials/payables/invoices/transactions",
      "REQUESTID": "455625",
      "STATUS": "SUCCEEDED"
    },
    {
      "JOBNAME": "Upload Interface Error and Job Output File to Universal Content Management",
      "JOBPATH": "/oracle/apps/ess/financials/commonModules/shared/common/interfaceLoader",
      "REQUESTID": "455629",
      "STATUS": "SUCCEEDED"
    }
  ],
  "SUMMARYSTATUS": "SUCCEEDED",
  "DOCUMENTID": "1691893"
}
```


Figure 29: Sample response from callback

Advanced Features

Securing the Inbound or Outbound Data File

Since inbound and outbound data files are transmitted over the Internet and often contain the company's sensitive information and financial transactions like journal entries, invoices, payments and bank records, data encryption is a critical and essential element in implementing your integrations with Oracle ERP Cloud. You can secure data files between Oracle ERP Cloud and your on-premise applications or systems including Platform as a Service (PaaS) applications. This topic describes how to set up and use encryption keys for secure file transfer. After you perform this setup, you can encrypt and decrypt files and transfer them between your servers and Oracle ERP Cloud using import and export bulk data processes.

Oracle ERP Cloud supports Pretty Good Privacy (PGP) unsigned encryption with 1024 bits key size. There are two types of encryption keys:

1. Oracle ERP Cloud PGP Key
2. Customer PGP Key

Oracle ERP Cloud PGP Key

A customer uses the public key to encrypt the inbound file. The import bulk data process will use the private key to decrypt the file before starting the load and import process. This key can be generated using the Security Console.

Customer PGP Key

A customer uses the private key to decrypt exported files from the Oracle ERP Cloud. The export bulk process will use the public key to encrypt the outbound file. The customer can import their public key into the Oracle ERP Cloud using the Security Console.

Note: Customers may use different keys for a different Cloud pod or the same key on multiple Cloud pods.

For more information on managing PGP keys, see *Appendix 13: Managing PGP Encryption Keys.*

Enabling Encryption

For Import Process

A customer encrypts inbound data file using the cloud public key. Oracle ERP Cloud decrypts this file using a cloud private key before starting the load and import process. These are the following steps to enable encryption in the import process:

1. Encrypt the data ZIP file using an Oracle ERP Cloud public key. To encrypt inbound data file, see *Appendix 14: How to Encrypt and Decrypt a Data File.*
2. In the payload for the importBulkData operation, specify the following job options:

Options	Value
FileEncryption	PGPUNSIGNED or PGPSIGNED
FA_ALIAS	Oracle ERP Cloud Key Alias Name
CUSTOMER_ALIAS	Customer Key Alias Name

Example: <typ:jobOptions>FileEncryption=PGPUNSIGNED,FA_ALIAS=<ERP_CLOUD_KEY>,CUSTOMER_ALIAS=<CUST OMER_KEY></typ:jobOptions>

Note: Alias names are defined when you generate an Oracle ERP Cloud key or import a customer key.

For Export Process

When enabled, Oracle ERP Cloud encrypts an extracted data file using a customer's public key and uploads the file to UCM. These are the following steps to enable encryption in the export process.

1. In the payload for the exportBulkData operation, specify the following job options:

Options	Value
FileEncryption	PGPUNSIGNED or PGPSIGNED
FA_ALIAS	Oracle ERP Cloud Key Alias Name
CUSTOMER_ALIAS	Customer Key Alias Name

Note: Alias names are defined when you generate an Oracle ERP Cloud key or import a customer key.

Decrypt the output file using the customer private key. To decrypt an outbound data file, see *Appendix 14: How to Encrypt and Decrypt a Data File.*

Job Property File for the Bulk Import Process

The Job Details parameter in the importBulkData operation includes the job definition and package names, as well as the job parameters of the imported object. To get the job package and definition name, see 'Viewing Details about Predefined Scheduled Processes' in the File-Based Data Import for Oracle Financials Cloud guide on the Oracle Help Center at http://docs.oracle.com. Use the following advanced options to specify the job details data associated with the importBulkData operation:

- Generate and add the Job Properties file to the data ZIP file
- Generate and upload the Job Properties file to the UCM applicable account for reusability

See *Appendix 11: Creating a Job Property File for the importBulkData Operation* for detailed information on how to generate the Job Property file.

Note

Parameter Precedence:

1. Payload Parameter File in the ZIP data file
2. Parameter File stored on UCM

Option 1: Job Property File as Part of the Data ZIP File

The following sample request payload illustrates the Journal Import process with the relevant parameter file, included together with the import data file in a ZIP file:

Sample Request Payload

```
<soap:Body>
       <ns1:importBulkData

       xmlns:ns1="http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrat
ionService/types/">
              <ns1:document

       xmlns:ns2="http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrat
ionService/">
                     <ns2:Content>
UEsDBBQAAAAIAKSUz0guTciD4QAAAKwDAAAPAAAAR2xJbnRlcmZhY2UuY3N2vZJBS8NAEIXvgv9hf8A0mZmdDbu9tSR60SK2
6jm0owRiIkk
9+O/dqDebSxc6h2UZvsfjPWZTvQABI/qcOLcEOx2P67qtu73Co773RzW7oe7GVx1ge1tGNHIoOQZYARIgIhARTp+fB+Zmcil
cZj0AsZBkRW
RvPtvW/PpFo68PNQtTrbfmTg9vOizNczM2fWf+tg86NP1haUrdL9BDsvz/BCbvAs9mqDbl9dXJ0sxca2V1n9JaxIIjDiGzkS
qYg0jATIrk9 Inyi5S3eko6OWudd
/HoJobFCqFkwsnRE+VnNvcNUEsDBBQAAAAIAIeUz0g8JMczbAAAAH4AAAAdAAAASm91cm5hbHNJbXBvcnRURVNULnBy
b3BlcnRpZXM1yDEKQjEMBuBd8A4eIBC7uDs4KEUE3wV+a6wP2qQkvsHbi4h822eO0oQxRrBE8GNWaJnRgquoOFqWexXn4VYd
PbhY76Z0ssU
V7diH+Stj0fIU/2/8ejpcJ0rbXaIL3rg1CUq0z5nOX5v16gNQSwECFAAUAAAACACklM9ILk3Ig+EAAACsAwAADwAAAAAAAAA
AACAAAAAAAA
AAR2xJbnRlcmZhY2UuY3N2UEsBAhQAFAAAAAgAh5TPSDwkxzNsAAAAfgAAAB0AAAAAAAAAAAAgAAAADgEAAEpvdXJuYWxzSW
1wb3J0VEVTV
C5wcm9wZXJ0aWVzUEsFBgAAAAACAAIAiAAAALUBAAAAAA==</ns2:Content>
                     <ns2:FileName>JournalsImportTEST_1234.zip</ns2:FileName>
              </ns1:document>
              <ns1:jobDetails></ns1:jobDetails>
              <ns1:notificationCode>30</ns1:notificationCode>
              <ns1:callbackURL>http://hostname:port/myCallbackService</ns1:callbackURL>
              <ns1:jobOptions></ns1:jobOptions>
       </ns1:importBulkData>
</soap:Body>
```

Figure 30 : Sample request payload for the Journals Import process with the parameter file comprehended along import data in a ZIP file

The following sample job property file for Journals Import (JournalsImportTEST.properties) is included with the import data file in a ZIP file:

```
oracle/apps/ess/financials/generalLedger/programs/common,JournalImportLauncher,JournalsI
mportTEST,1061,Payables,1,ALL,N,N,N
```

Figure 31: Sample parameter file for Journals Import

See *Appendix 11: Creating a Job Property File for the importBulkData Operation* for detailed information on creating the job properties file.

Option 2: Upload the Job Properties File to UCM for Reuse

After the file is uploaded to the UCM applicable account, there are two options to reuse the file:

1. Add JobDetailFileName=<FileName.properies> in <jobOptions>
2. Follow file naming convention as defined in *Appendix 11: Creating a Job Property File for the importBulkData Operation.*

The following sample payload illustrates the Journal Import process with the job properties file uploaded to UCM. The parameter file should be uploaded using the specific UCM account associated with a particular import process:

Sample Request Payload

```
<soap:Body>
       <ns1:importBulkData

       xmlns:ns1="http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrat
ionService/types/">
              <ns1:document

       xmlns:ns2="http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrat
ionService/">
                     <ns2:Content>

UEsDBBQAAAAIAKSUz0guTciD5gAAAKwDAAAPAAAAR2xJbnRlcmZhY2UuY3N2tZLBasMwDIbvg72DHkB1JFkJcW8tyXbZwliz
7hxabwSyZCT
dYW8/Z/SWFgZm/8EY+/sl9KOqfEVGIcoTlsQy1n46bZuu6Q8en/3HcPJQj00/vfkRd/dFQANHmpDDDRIjESEz03z5PfCqQpc
sNTZHZFFWkw
X27qvr4NwP6u9PDysotzt48Md3P65h307t0MP59cmP7XBcQ+EPKwp1Yu1LOeE8dXLxb1ZZFbc31aXQ4FpqRfkYk1rAXMrinL
GBykScqiOjW
fT0kfal/iG8zUvUylmb5mlYupkRtcqkRiV69Ej7Un9K7gdQSwECFAAUAAAACACklM9ILk3Ig+YAAACsAwAADwAAAAAAAABA
CAAAAAAAAAA

R2xJbnRlcmZhY2UuY3N2UEsFBgAAAAABAAEAPQAAABMBAAAAAA==</ns2:Content>
                     <ns2:FileName>JournalsImportTEST_1234.zip</ns2:FileName>
              </ns1:document>
              <ns1:jobDetails></ns1:jobDetails>
              <ns1:notificationCode>30</ns1:notificationCode>

JobDetailFileName=JournalsImportTEST.properties


              <ns1:callbackURL>http://hostname:port/myCallbackService</ns1:callbackURL>
              <ns1:jobOptions></ns1:jobOptions>
       </ns1:importBulkData>
</soap:Body>
```

Figure 32: Sample request payload for the JournalsImport process with the parameter file uploaded to UCM

Specifying Multiple Threads in Bulk Import

To increase the throughput when importing data, users can specify multiple threads in the import process. It supports a maximum of 10 threads for sequential processing and a maximum of 5 threads for parallel processing. After the data file is loaded in the interface table, the import process will start batch processing based on number of job parameters records defined in a property file and job option in the payload. The default option is sequential and the following property in <jobOptions> attribute could enable parallel processing:

<jobOptions>ExecutionMode=Parallel<jobOptions>

In a job property file, you must enter multiple records with parameter values that can import data in batches either sequentially or concurrently. In the sequential pattern, an import process stops when a batch fails. The remaining batch processes will not be executed, and callback will include all the details including the failed process.

The following is the sample property file of journals import where data file contains 3 different ledgers:

oracle/apps/ess/financials/generalLedger/programs/common,JournalImport Launcher,GL,1061,Payables,**1**,ALL,N,N,N

oracle/apps/ess/financials/generalLedger/programs/common,JournalImport Launcher,GL,1061,Payables,**2**,ALL,N,N,N
oracle/apps/ess/financials/generalLedger/programs/common,JournalImport

Launcher,GL,1061,Payables,**3**,ALL,N,N,N

ERP will import three ledgers either sequentially or concurrently depending on the “ExecutionMode” type. The job package and name must be same for all the records.

Optimized Management of Large Data Files

The Oracle ERP Integration Service provides the capability to attach data files instead of converting data files to base64 encoding. The attachment feature leverages the Message Transmission and Optimization Mechanism (MTOM) approach by reducing the request payload size as file content is not part of the payload (in base64 encoding). This process optimizes the handling of large files for both inbound and outbound processes. You need minor changes in your web service proxy code to enable MTOM support.

For more information about the MTOM changes, see *Appendix 15: Large File Optimization (MTOM) Proxy Client Code Changes.*

Appendix 1: Security Prerequisites to Download the Job Output File

The ESS job log and output files are placed in the Attachments Security group under the Oracle Universal Content Management server (Oracle WebCenter Content server). You must have access to the security group called Attachments to download the log file or the output file with the ERP Integration Service.

This access can be granted via the security role called AttachmentsUser.

Use the Security Console to grant access to the AttachmentsUser role. The Security Console can be accessed in the following ways:

- Use the Manage Job Roles or Manage Duties tasks in the Setup and Maintenance work area.
- Select **Navigator** - **Tools** - **Security Console**

Access to the Security Console is provided by the predefined IT Security Manager role.

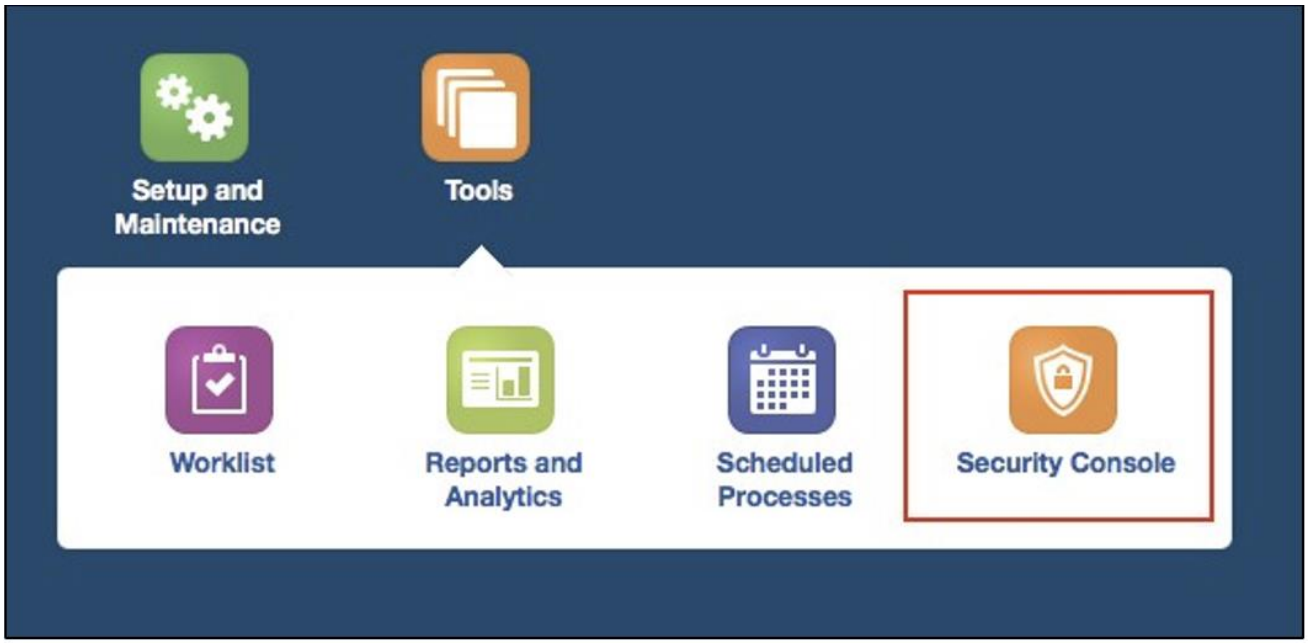


Figure 33: Steps to access Security Console from the Navigator

The AttachmentsUser role is inherited by the predefined Employee and Contingent Worker roles. You can verify this inheritance by querying the role AttachmentsUser from the Security Console and use the Expand Toward **Users** and show the **Roles** option.

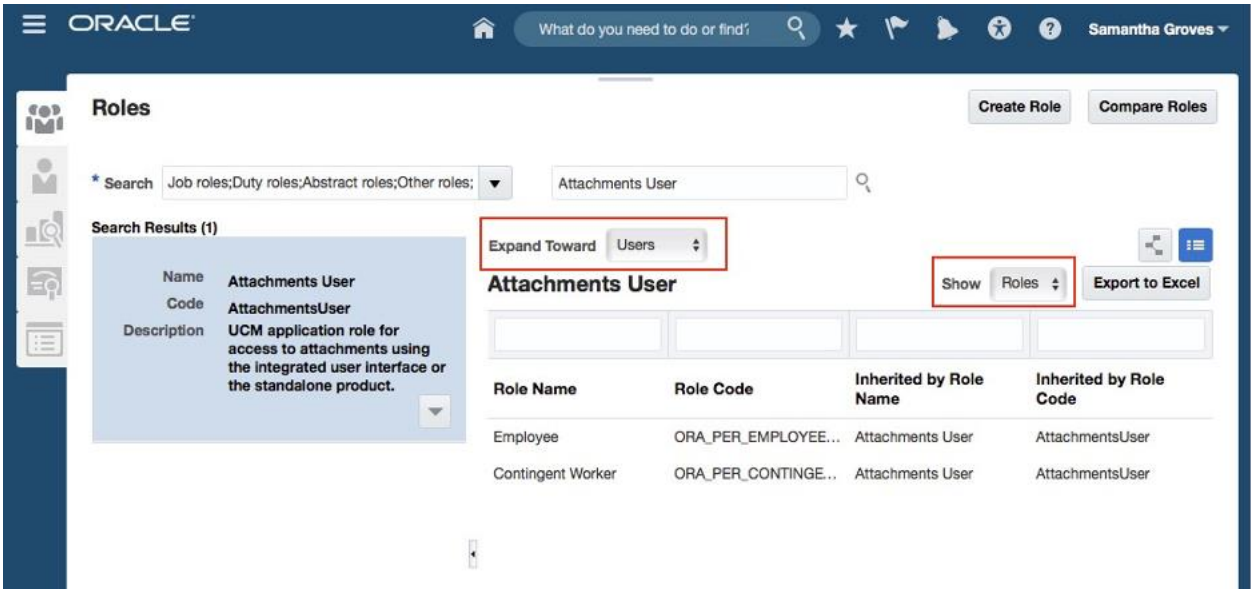


Figure 34: Screenshot to verify inheritance of 'Attachment User' role.

After reviewing the role inheritance of the AttachmentsUser role, review the users that are currently assigned the AttachmentsUser role.

You can verify role assignments to users by querying the role AttachmentsUser from the Security Console and use the Expand Toward **Users** and show **Users** option.

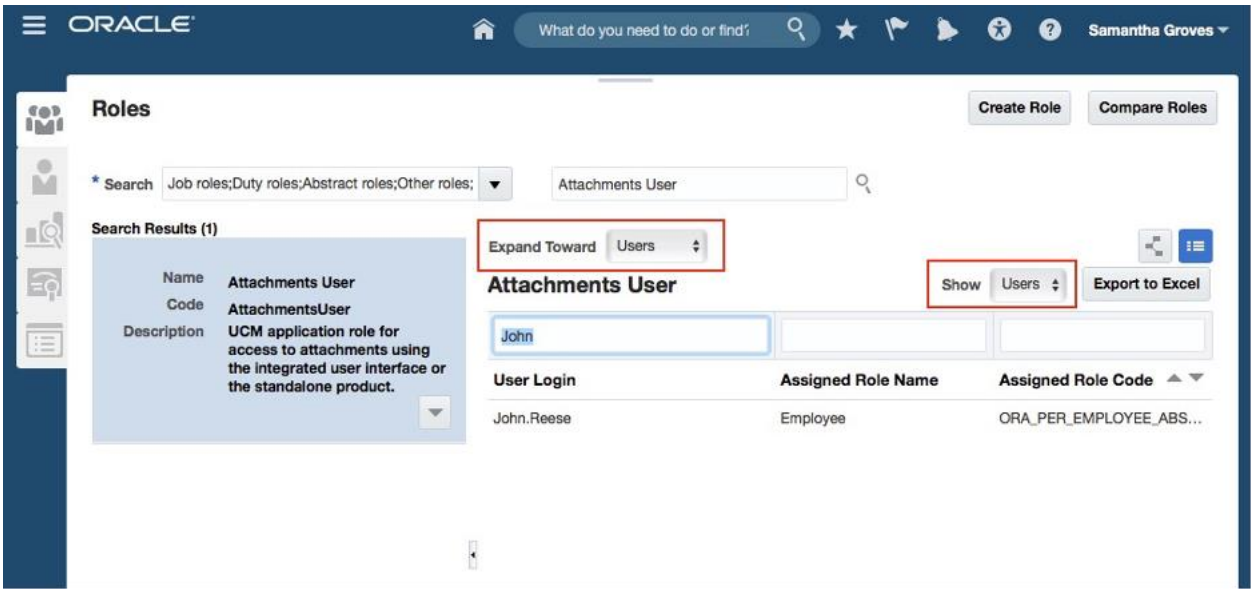


Figure 35: Screenshot to verify user assignments to Attachments User role

In above figure, the user John.Reese have been assigned the AttachmentsUser role through the predefined Employee role.

Lastly, verify that the Attachments security group is listed in the UCM Search page.

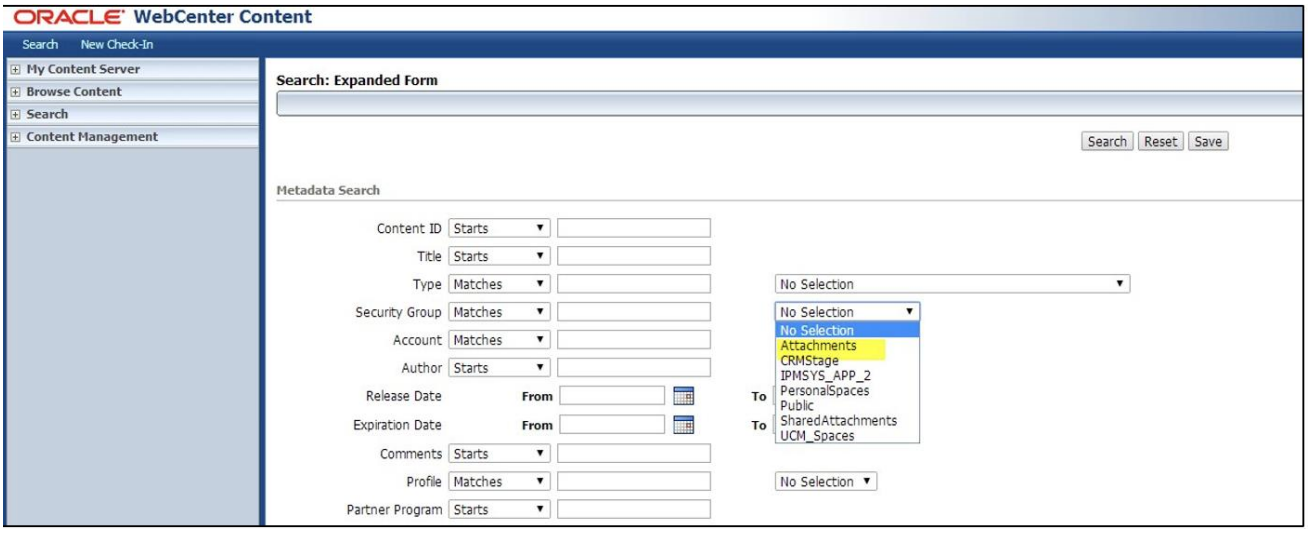


Figure 36: Search page for UCM to verify user's access to Attachments security group.

Appendix 2: Sample Code for Preparing a Data File for Inbound and Outbound Flow

The following example illustrates sample code for preparing a data file for the inbound flow.

Sample File Name: UtilEncodeBase.java

```
import java.io.ByteArrayOutputStream;
import java.io.File;

import java.io.FileInputStream;
import java.io.FileReader;
import java.io.IOException;
import java.io.InputStream;

import org.apache.commons.codec.binary.Base64;

public class UtilEncodeBase {
```

```
    public UtilEncodeBase() {
        super();
    }

    public static void main(String[] a) throws Exception {
        // Enter the filename as input         File br = new File(a[0]);         // Convert the
file into Byte          byte[] bytes = loadFile(br);
        // Call the api for Base64 encoding

        byte[] encoded = Base64.encodeBase64(bytes);

        String encStr = new String(encoded);

        // Print the file

        System.out.println(encStr);
    }

private static byte[] getByteArray(String fileName) {

        File file = new File(fileName);

        FileInputStream is = null;

        ByteArrayOutputStream buffer = new ByteArrayOutputStream();
        int nRead;

        byte[] data = new byte[16384];
        try {

            is = new FileInputStream(file);

            while ((nRead = is.read(data, 0, data.length)) != -1) {
                buffer.write(data, 0, nRead);

            }

            buffer.flush();

        } catch (IOException e) {

            System.out.println("In getByteArray:IO Exception");

            e.printStackTrace();
        }

        return buffer.toByteArray();

    }
    private static byte[] loadFile(File file) throws IOException {
        InputStream is = new FileInputStream(file);

        long length = file.length();
        if (length > Integer.MAX_VALUE) {

            // File is too large
        }

        byte[] bytes = new byte[(int) length];

        int offset = 0;
        int numRead = 0;

        while (offset < bytes.length &&

            (numRead = is.read(bytes, offset, bytes.length - offset)) >=

            0) {
```

```
            offset += numRead;
        }

        if (offset < bytes.length) {

            throw new IOException("Could not completely read file " + file.getName());

        }
        is.close();
        return bytes;
    }
}
```

Figure 37: Sample code for the inbound data flow

The following example illustrates sample code for preparing a data file for the outbound flow.

Sample FileName: UtilDecodeBase.java

```
import java.io.ByteArrayOutputStream;
import java.io.File;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.FileReader;
import java.io.IOException;
import java.io.InputStream;

import org.apache.commons.codec.binary.Base64;


public class UtilDecodeBase {
    public UtilDecodeBase() {
        super();
    }

    public static void main(String[] a) throws Exception {
        System.out.println("Start");
                // Read the inputsteam

        String encStr = a[0];


                // Run the api to perform the decoding


        byte[] rbytes = Base64.decodeBase64(encStr.getBytes());

                // Put the location for the output file


        FileOutputStream os = new FileOutputStream("/tmp/Test1234.zip");
        os.write(rbytes);
        os.close();
    }

}
```

Figure 38: Sample code for the outbound data flow

Appendix 3: Predefined Target UCM Accounts

You can transfer data files to predefined accounts in the Oracle WebCenter Content server (UCM) that correspond to the interface table.

To find the UCM account:

1. Open the File Based Data Import guide for your cloud service.
2. Locate your respective import process. For example, **Journal Import**.
3. View the UCM account in the Details section.

Appendix 4: ESS Job Execution Status Monitoring

Use the getEssJobStatus operations to monitor the status of an ESS job. Invoke the getEssJobStatus operation in loop until either a terminal status is received in response or maximum retry is reached. The terminal statuses are SUCCEEDED, CANCELLED, ERROR, and WARNING.

Terminal ESS Job Execution Status	Description	User Action
SUCCEEDED	Request completed and was successful.	Check the details of the completed process and proceed with any post processing.
ERROR	Request processed but resulted in error.	Download the details of the error and correct the data.
WARNING	Request processed but resulted in a warning.	Download the details of the process. Check the reason for the warnings and take the necessary action to correct the input data.
CANCELED	Request was canceled.	Resubmit the request if required.

Appendix 5: Testing Web Service using a client Proxy

Perform the following steps to test a web service operation using JDeveloper:

1. Import a new certificate in the keystore.
2. Create a web service client proxy and add the OWSM policy.
3. Test the web service.

Steps to Import a new certificate in the Keystore:

1. Export the certificate from the browser to the file, using the following steps:
    a. Access the SSL URL for any web service using browser..
    b. On Internet Explorer, click **Tools** > **Internet Options**.
    c. On the **Content** tab, click **Certificates**. On the **Personal** tab, select the **Baltimore CyberTrust Root** certificate and click **View**. The certificate hierarchy appears; export the top two certificates (**Baltimore CyberTrust Root** and **Verizon Akamai SunServer CA G14-SHA1**).

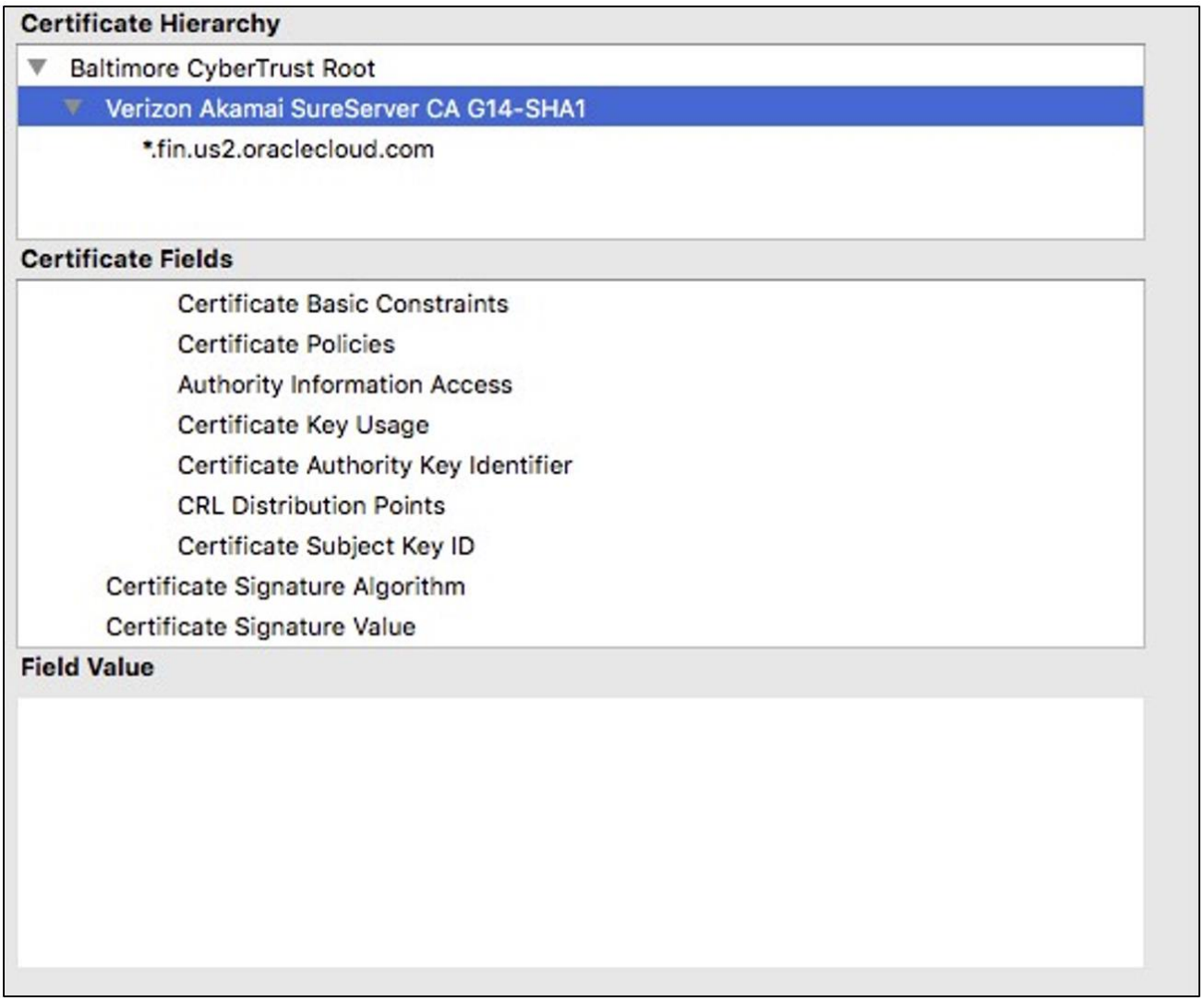


Figure 39: Screenshot to select the certificate

d. On the **Certification Path** tab, select **Baltimore CyberTrust Root** and click **View Certificate**.

e. On the **Details** tab, select **Copy to File**. The Certificate Export Wizard appears.

f. Click **Next** > **Next** and enter a name and location for the file you want to export.

g. Change the encoding to Base-64 and click **Next**.

h. Provide the file name and click **Finish**.

i. Repeat steps iv to viii for the **Verizon Akamai SunServer CA G14-SHA1** certificate.

When using other web browsers, perform similar steps. The navigation path may differ in the web browsers used.

2. Type the following command to import a certificate into keystore:

   keytool -import -trustcacerts -file <filename> -alias <aliasname> -keystore defaultkeystore.jks - storepass welcome1

3. Run the following command to verify if the trust store contains the imported certificates:

   keytool -list -v -keystore <filename> -storepass welcome1 | grep -i Verizon

Create a Proxy Client and Add the OWSM Policy

1. Create a new project and select **Web Services Proxy**.

2. Set the client style to **JAX-WS Style**.
3. Select the web service description, for example, *https://<Hostname>.<Domain Name>:<Port No>/fscmService/ErpIntegrationService?WSDL.*
4. Select the **Copy WSDL Info Project** check box. Specify the default mapping options.
5. Specify the asynchronous method.
6. Select the **Do not generate any asynchronous methods.**
7. Click **Finish**.
8. Once the proxy client code is generated, add the following variables:

    - jksFPath: File location that has the certificate to add to the keystore. For example, *D:\fintuilwdestapp\Project5\client.jks*
    - jksPassword: Password to access WSDL. For example, **Welcome1**.
    - trustStore: Path where the certificates are stored, used during java installation by default.
    - trustStorePassword: Password for truststore.
    - Username: User name to sign in to the service.
    - Password: Password for the user to sign in to the service. For example, **Welcome1**.
    - endpointNonSSLURL: URL for the FinUtilService service.
    - serviceName: Schema of the service used to add the policies.
    - securityFeature: Policy used to add to the service.

    Note: An example of a message protection policy is policy:oracle/wss_username_token_over_ssl_client.

9. Create the invokeServiceWithUsernameTokenMessageProtectionPolicy() method to add policy.

Test Upload File to UCM using Web Service

To test the file upload to the UCM server:

1. Create a sample payload associated with the uploadFileToUcm operation.
2. Create the method *invokeUpload* to call the operation uploadFileToUcm.

Export the Certificate

To export the certificate associated with the web service from the browser, invoke the end point URL for the web service

https://<hotaname>.<domainname>/fscmService/ErpIntegrationService?WSDL.

1. Copy the content from the XML element *dsig:X509Certificate.*

Figure 40: Sample content of the dsig:X509Certificate

2. To use the key contained in this certificate, create a local KeyStore and import the certificate into it:
    -
    - Create a new file with any name you like. You must change the extension to .cer to indicate that it is a certificate file.
    - Using a text editor, open the file you just created and enter "----BEGIN CERTIFICATE----"on the first line.
    - In the next line, copy the Base64 encoded certificate from the service WSDL file to the newly created certificate file.
    - Add "-----END CERTIFICATE-----" on a new line and save the file. Now you have a certificate containing the public key from the server.
    - Open the command line and change the directory to $JAVA_HOME/bin. Use the following command to create a KeyStore and import the public key from the certificate:

Figure 41: Sample certificate file (<Filename>.cer)

3. Add the variables to the proxy client code.

ErpIntegrationServiceSoapHttpPortClient.java

```
public class ErpIntegrationServiceSoapHttpPortClient

{

   @WebServiceRef

   private static ErpIntegrationService_Service ErpIntegrationService_Service;
   private static final AddressingVersion WS_ADDR_VER = AddressingVersion.W3C;
```

```
    // Add the additional variables


    private final String jksFPath = "D:\\fintuilwdestapp\\Project5\\client.jks";
    private final String jksPassword = "Welcome1";


    private final String trustStore = "C:\\ProgramFiles\\Java\\jdk1.7.0_51\\jre\\lib\\security\\cacerts";
    private final String trustStorePassword = "";
    private final String username = "finuser1";
    private final String password = "Welcome1";


    private String endpointNonSSLURL = "https://efops-rel9st1-cdrm1-external-
fin.us.oracle.com/fscmService/ErpIntegrationService";
    private static final QName servicename = new

    QName("http://xmlns.oracle.com/apps/financials/commonModules/shared/ErpIntegrationService/", "ErpIntegra
tionService");


    private SecurityPolicyFeature[] securityFeature = new SecurityPolicyFeature[] {
        new SecurityPolicyFeature("policy:oracle/wss_username_token_over_ssl_client_policy")
    };
    private ErpIntegrationService ErpIntegrationService;

    // End add the additional variables public static void main(String [] args)   {  System.out.println("inside main");


    ErpIntegrationServiceSoapHttpPortClient f = new ErpIntegrationServiceSoapHttpPortClient();


    f.invokeServiceWithUsernameTokenMessageProtectionPolicy();


    String retStatus = f.invokeUpload();

        //long submittedJobId = f.invokeSubmitJob(retStatus);

        //f.invokeEssJobStatus(submittedJobId);

        //f.invokeDownloadESSJobExecDetails(submittedJobId);


}
```

Figure 42: Sample proxy code with variables for ERPIntegration SOAP service

4. Create the invokeServiceWithUsernameTokenMessageProtectionPolicy()method to add the policy.

invokeServiceWithUsernameTokenMessageProtectionPolicy()

```
public void invokeServiceWithUsernameTokenMessageProtectionPolicy() {


        System.out.println("inside invokeservice");

        URL wsdlDoc = null;
        try {

            wsdlDoc = new URL("https://efops-rel9st1-cdrm1-external-
```

```
        fin.us.oracle.com / fscmService / ErpIntegrationService ? wsdl ");
         }catch(MalformedURLException e){

        e.printStackTrace();

      }

      System.setProperty("javax.net.ssl.trustStore", trustStore);


      System.setProperty("javax.net.ssl.trustStorePassword", trustStorePassword);


      ErpIntegrationService_Service = new ErpIntegrationService_Service(wsdlDoc, servicename);
      ErpIntegrationService =

        ErpIntegrationService_Service.getErpIntegrationServiceSoapHttpPort(securityFeature);


      WSBindingProvider wsbp = (WSBindingProvider) ErpIntegrationService;
      Map < String, Object > requestContext = wsbp.getRequestContext();
      requestContext.put(BindingProvider.USERNAME_PROPERTY, username);
      requestContext.put(BindingProvider.PASSWORD_PROPERTY, password);


      requestContext.put(BindingProvider.ENDPOINT_ADDRESS_PROPERTY, endpointNonSSLURL);
      requestContext.put(ClientConstants.WSSEC_KEYSTORE_TYPE, "JKS");
      requestContext.put(ClientConstants.WSSEC_KEYSTORE_LOCATION, jksFPath);
      requestContext.put(ClientConstants.WSSEC_KEYSTORE_PASSWORD, jksPassword);
      System.out.println("Finished invokeservice");

    }
```

Figure 43: Sample method code to add the policy for ErpIntergration SOAP service

ErpIntergration SOAP Service

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">   <soap:Body>

    <ns1:uploadFileToUcm

xmlns:ns1="http://xmlns.oracle.com/apps/financials/commonModules/shared/ErpIntegrationService/types/">

      <ns1:document

xmlns:ns2="http://xmlns.oracle.com/apps/financials/commonModules/shared/ErpIntegrationService/">          <ns
2:Content>UEsDBBQAAAAIAEhrrkQSDhHq5BLBQYAAAAAAQABAD0AAAATPQEAAAA=</ns2:Content>

        <ns2:FileName>TestUploadFileName.zip</ns2:FileName>          <ns2:ContentType>zip</ns2:ContentType>

        <ns2:DocumentTitle>Sample File1</ns2:DocumentTitle>

        <ns2:DocumentAuthor>finuser1</ns2:DocumentAuthor>

        <ns2:DocumentSecurityGroup>FAFusionImportExport</ns2:DocumentSecurityGroup>

        <ns2:DocumentAccount>fin$/generalLedger$/import$</ns2:DocumentAccount>

      </ns1:document>
```

```
    </ns1:uploadFileToUcm>

  </soap:Body>

</soap:Envelope>
```

Figure 44: Sample payload for the uploadFileToUCM operation

invokeUpload Method

```
private String invokeUpload()  {


   System.out.println("inside invokeupload");
   ObjectFactory objectFactory = new ObjectFactory();

   DocumentDetails documentDet = new DocumentDetails();

   String fileNameWithPath = "C:\\Users\\NGARLAPA\\Desktop\\Sample.zip";

   String fileName = "Sample.zip";

   String contentType = "zip";

   String title = "Journals Import";

   String ucmAccountInfo = "fin$/generalLedger$/import$";
   String ucmSecurityGroup = "FAFusionImportExport";

   documentDet.setContent(objectFactory.createDocumentDetailsContent(getByteArray(fileNameWithPath)));
   documentDet.setContentType(objectFactory.createDocumentDetailsContentType(contentType));

   documentDet.setDocumentAccount(objectFactory.createDocumentDetailsDocumentAccount(ucmAccountInfo));
   documentDet.setDocumentAuthor(objectFactory.createDocumentDetailsDocumentAuthor(username));


documentDet.setDocumentSecurityGroup(objectFactory.createDocumentDetailsDocumentSecurityGroup(ucmSecuri
tyGroup));
   documentDet.setDocumentTitle(objectFactory.createDocumentDetailsDocumentTitle(title));
   documentDet.setFileName(objectFactory.createDocumentDetailsFileName(fileName));


   UploadFileToUcm uploadFileToUcm = new UploadFileToUcm();


   uploadFileToUcm.setDocument(documentDet);
   UploadFileToUcmResponse retStatus = null;
   try {


      retStatus = ErpIntegrationService.uploadFileToUcm(uploadFileToUcm);


      System.out.println("File successfully Uploaded.Status is:" + retStatus.getResult());
   } catch (Exception e) {


      e.printStackTrace();
   }


   return retStatus.getResult();
```

```
}

private byte[] getByteArray(String fileName) {

   File file = new File(fileName);

   FileInputStream is = null;

   ByteArrayOutputStream buffer = new ByteArrayOutputStream();
   int nRead;

   byte[] data = new byte[16384];
   try {

      is = new FileInputStream(file);

      while ((nRead = is.read(data, 0, data.length)) != -1) {
         buffer.write(data, 0, nRead);
      }

      buffer.flush();

   } catch (IOException e) {

      System.out.println("In getByteArray:IO Exception");

      e.printStackTrace();
   }

   return buffer.toByteArray();

}
```

Figure 45: Sample payload for method invokeUpload() to invoke the uploadFileToUCM operation

Appendix 6: Automate Web Service Invocation Using JDeveloper 11

The automation approach for the web service invocation includes the following:

- Compile the web service client proxy project created in *Appendix 5: Testing Web Service using a Client Proxy*.
    - Add the following JDeveloper 11g jars to the deployment profile:
        - weblogic.jar : *$MW_HOME/wlserver_10.3/server/lib/weblogic.jar*
        - jrf.jar**:** *$MW_HOME/oracle_common/modules/oracle.jrf_11.1.1/jrf.jar*

- Create the deployment profile for the project:
  - Click **Project Properties** > Click **Deployment** > Click **New** > Select **Archive Type** as JAR File and specify the name.
  - Select the **Include Manifest File** (META-INF/MANIFEST.MF) option and specify the value for Main Class.

For example, *oracle.apps.finacial.testUtil.proxy.client.ErpIntegrationServiceSoapHttpPortClient*

- Generate the jar file and execute the following command. For example, *java –classpath $CLASSPATH –jar <JAR File Name>*

Appendix 7: Error Handling for Import Jobs

To address errors generated during the import process, use the following methods:

- If an ADFdi correction spreadsheet is available, use the spreadsheet to correct the data in the interface table and resubmit the import process.
- If no correction spreadsheet is available, use the purge process to delete all the records from the interface table. Correct the data in the original data file and upload the file again using the correct UCM account.

Repeat the process until your data is successfully imported.

Error Handling Processes

The following table lists the existing error handling processes which may be used to address any errors encountered during the import process for each respective product interface table.

Import Process	Correction Spreadsheet	Steps
Import Payables Invoices	CorrectImportErrors.xlsx	In the Invoices work area, navigate to Correct Import Errors in the Tasks region.
Import AutoInvoice	ManageInvoiceErrors Spreadsheet.xlsx	In the Billing work area, navigate to the Review AutoInvoice Errors table. Click **Number of Errors**. Select the **Manage AutoInvoice Lines** spreadsheet.
Process Receipts through Lockbox	ManageLockboxErrors.xlsx	In the Receivables work area, navigate to Receivable Balances.
Fixed Asset Mass Additions Import	PrepareSourceLines.xlsx	In the Fixed Assets work area, navigate to Additions. Select **Pending Source Lines**.
Fixed Asset Mass Adjustments Import	UpdateMassFinancialTrans action.xlsm	In the Fixed Assets work area, navigate to Financial Transactions. Select **Pending Mass Financial Transactions**.
Fixed Asset Mass Retirements Import	UpdateMassRetirements.xls m	In the Fixed Assets work area, navigate to Retirements. Select **Pending Retirements**.

Fixed Asset Mass Transfers Import	UpdateMassTransfers.xlsm	In the Fixed Assets work area, navigate to Pending Mass Transfers. Select **Pending Mass Transfers**.
Journal Import	JournalCorrections.xlsx	In the Journals work area, navigate to Correct Import Errors in the Tasks region.

Appendix 8: Manage Inbound Flow Automation Steps with Separate Web Service Operations

Inbound data integration is achieved seamlessly using the importBulkData operation from the Oracle ERP Integration Service. However, for certain use cases, you may want to control the flow and orchestration using separate operations.

The following section describes how to import journals using the individual web service operations. The same process applies to all the supported FBDI objects.

Perform the following steps to control the orchestration flow using individual operations:

1. Generate the CSV file as previously outlined.
2. Use the following operations to control the orchestration flow:
    a. Upload the file using uploadFileToUcm operation. See the *UploadFileToUcm Request Payload in Inbound Management Section.*
    b. Submit the Load Interface File for Import job using the submitESSJobRequest operation(load to interface tables). See *SubmitESSJobRequest Request Payload.*
        i. The Load Interface File for Import job takes four parameters submit.argument1, submit.argument2, submit.argument3 and submit.argument4 which corresponds to Interface Option Id, Document Id of Data File , constant string "N" and constant string "N".
        ii. To get details of Interface Option Id, Refer *Appendix 21: Supported Import Processes by Load Interface File for Import Job(FBDI)*
        iii. Document Id of Data File is obtained in response of step a.
    c. Monitor the Load Interface File for Import job till completion using getEssJobStatus operation, Refer *GetESSJobStatus Response Payload and Appendix 4: ESS Job Execution Status Monitoring*
    d. Submit the Import Journal job using submitESSJobRequest operation(load from interface tables to application tables), Refer *SubmitESSJobRequest Request Payload.*
        i. To get details of Import Job, Refer *Appendix 21: Supported Import Processes by Load Interface File for Import Job(FBDI)*
    e. Monitor the Import Journal job till completion using getEssJobStatus operation, Refer *GetESSJobStatus Response Payload and Appendix 4: ESS Job Execution Status Monitoring*
    f. Use downloadESSJobExecutionDetails operation to download the LOG and OUT files of requests submitted in step b and d.

Appendix 9: Manage Outbound Flow Automation Steps with Separate Web Service Operations

This section describes an alternate way to automate outbound data integration using individual web service operation. The Receivable Billing History Extract flow illustrates the outbound data integration flow with Oracle ERP Cloud. In this example, a batch of transactions are extracted from the system and sent to the customers. The transactions are extracted in the output file of the Receivables Billing History Extract ESS process.

Use the following operations to control the orchestration of the outbound flow:

1.
    a. Submit the Receivable Billing History Extract using the submitESSJobRequest operation. See *SubmitESSJobRequest Request Payload.*
    b. Monitor the Receivable Billing History Extract job till completion using the getEssJobStatus operation. See *GetESSJobStatus Response Payload and Appendix 4: ESS Job Execution Status Monitoring*
    c. Use downloadESSJobExecutionDetails operation to download the LOG and OUT files of requests submitted in step b and d.

Caution: Due to the synchronous nature of the downloadESSJonExecutionDetails operation, it is not recommended to use the operation for downloading large report files. Instead use ExportBulkData for Outbound Integration as described in the *Outbound Data Management section.*

Appendix 10: Creating a Callback Web Service

Bulk import and export operations in ERP integration services are long running processes that require asynchronous callback pattern to notify consumer upon job completion/error. ERP Cloud integration services provides infrastructure for customer to register their callback service that could be invoked when import or export job is completed. Customer may create and host a callback web service either in Oracle PaaS or on-premise application server to receive notification from ERP cloud upon job completion.

Implementation Consideration

In practice, customers will create and host a callback web service to optimally leverage the callback capabilities provided by Oracle ERP Integration Service for notification purposes. The customer callback web service must implement the onJobCompletion() operation. When a job completes, Oracle ERP Integration Service invokes the customer callback web service as defined in the request payload of supported operations with callback capabilities, such as the importBulkData, exportBulkData, submitJobWithOutput operations.

Oracle ERP Cloud will invoke this service upon job completion respectively. You can leverage any Oracle PaaS solution or other Third Party Solutions to host callback service.

Note: The customer callback web service triggers when the last job in the payload is executed, irrespective of whether the job completes successfully or fails.

Sample callback response


```
{
 "JOBS": [
  {
   "JOBNAME": "Load Interface File for Import",
   "JOBPATH": "/oracle/apps/ess/financials/commonModules/shared/common/interfaceLoader",
   "DOCUMENTNAME": "GlInterface__1.csv",
   "REQUESTID": "43648",
   "STATUS": "SUCCEEDED",
   "CHILD": [
    {
     "JOBNAME": "Load File to Interface",
     "JOBPATH": "/oracle/apps/ess/financials/commonModules/shared/common/interfaceLoader",
     "REQUESTID": "43650",
     "STATUS": "SUCCEEDED"
    },
    {
     "JOBNAME": "Transfer File",
     "JOBPATH": "/oracle/apps/ess/financials/commonModules/shared/common/interfaceLoader",
```

```
        "REQUESTID": "43649",
        "STATUS": "SUCCEEDED"
      }
    ]
  },
  {
    "JOBNAME": "Upload Interface Error and Job Output File to Universal Content Management",
    "JOBPATH": "/oracle/apps/ess/financials/commonModules/shared/common/interfaceLoader",
    "REQUESTID": "43651",
    "STATUS": "SUCCEEDED"
  },
  {
    "JOBNAME": "Purge Interface Tables",
    "JOBPATH": "/oracle/apps/ess/financials/commonModules/shared/common/interfaceLoader",
    "REQUESTID": "43652",
    "STATUS": "SUCCEEDED"
  }
 ],
 "SUMMARYSTATUS": "SUCCEEDED",
 "DOCUMENTID": "29639"
}
```


Figure 46: Sample callback response.

The following example illustrates a sample Java code for retrieving the details from the response payload:

Callback Service Implementation Sample Code

```
public void onJobCompletion(OnJobCompletion params) {

  String strPayload = null;
  strPayload = params.getResultMessage();
  JSONObject jsonResponse = new JSONObject(strPayload);
  String xmlStr = XML.toString(jsonResponse);

  // Use the xml parser api to retrieve the value of the node

  try{
     DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
     DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
     InputSource is = new InputSource();
     is.setCharacterStream(new StringReader(xmlStr));
     Document doc = dBuilder.parse(is);

     System.out.println("Root element :" + doc.getDocumentElement().getNodeName());
     if (doc.hasChildNodes()) {
        NodeList nodeList = doc.getChildNodes();
        for (int count = 0; count < nodeList.getLength(); count++) {
           Node tempNode = nodeList.item(count);
           if (tempNode.getNodeType() == Node.ELEMENT_NODE) {
              if (tempNode.getNodeName().equals("DOCUMENTID")) {
                 System.out.println("DOCUMENTID: " + tempNode.getTextContent());
              }
              if (tempNode.getNodeName().equals("SUMMARYSTATUS")) {
                 System.out.println("SUMMARYSTATUS: " + tempNode.getTextContent());
              }

              if (tempNode.getNodeName().equals("JOBS") && tempNode.hasChildNodes()) {
                 NodeList jobList = doc.getChildNodes();
                 for (int c = 0; c < jobList.getLength(); c++) {
                    printJobDetails(job);
```

```
                    //printing child
                    if (job.hasChildNodes()) {
                        NodeList nodeList2 = job.getChildNodes();
                        for (int count2 = 0; count2 < nodeList2.getLength(); count2++) {
                            Node tempNode2 = nodeList2.item(count2);
                            if (tempNode2.getNodeType() == Node.ELEMENT_NODE && tempNode2.hasChildNodes()) {
                                NodeList nodeList3 = tempNode2.getChildNodes();
                                for (int count3 = 0; count3 < nodeList3.getLength(); count3++) {
                                    printJobDetails(nodeList3.item(count3));
                                }
                            }
                        }
                    }
                }
            }
        }
    }
  }catch(Exception e) {
  e.printStackTrace(); }

private void printJobDetails(Node job) {
  if (job.hasChildNodes()){
    NodeList nodeList = job.getChildNodes();
    for (int count = 0; count < nodeList.getLength(); count++) {
      Node tempNode = nodeList.item(count);
      if (tempNode.getNodeType() == Node.ELEMENT_NODE && !tempNode.hasChildNodes()) {
        if (tempNode.getNodeName().equals("JOBNAME")) {
          System.out.println("JOBNAME: " + tempNode.getTextContent());
        }
        if (tempNode.getNodeName().equals("JOBPATH")) {
          System.out.println("JOBPATH: " + tempNode.getTextContent());
        }
        if(tempNode.getNodeName().equals("STATUS")) {
          System.out.println("STATUS        : " + tempNode.getTextContent());
        }
        if (tempNode.getNodeName().equals("REQUESTID")) {
          System.out.println("REQUESTID: " + tempNode.getTextContent());
        }
        if (tempNode.getNodeName().equals("DOCUMENTNAME")) {
          System.out.println("DOCUMENTNAME: " + tempNode.getTextContent());
        }
      }
    }
  }
}
```

Figure 47: Sample code for the callback service

Web Service Security & Pre-requisites

- Oracle ERP cloud implements Oracle Web Service Manager (OWSM) to secure web services.
- The security policy for callback is: oracle/wss_saml_token_bearer_client_policy.
- This mandates that the callback web service in your PaaS or on-premise must be secured through compatible OWSM server policy: oracle/wss_saml_bearer_or_username_token_service_policy.
- ERP callback implements SAML bearer token policy. In order for Client to receive callback, the issuer name (username) used in SAML assertion must exist in the Client identity store.
- The issuer name is the username used when invoking Inbound (importBulkData) or Outbound (exportBulkData) must be present in the IDM of PaaS Service application. The password for the username need not be the same.

- Oracle ERP Cloud certificates must be imported on the client side (where Callback is deployed) to authenticate SAML assertion from Oracle ERP cloud.

Note: Oracle ERP Cloud does not import the self-signed SSL certificate to Keystore.

Certificates Retrieval Steps

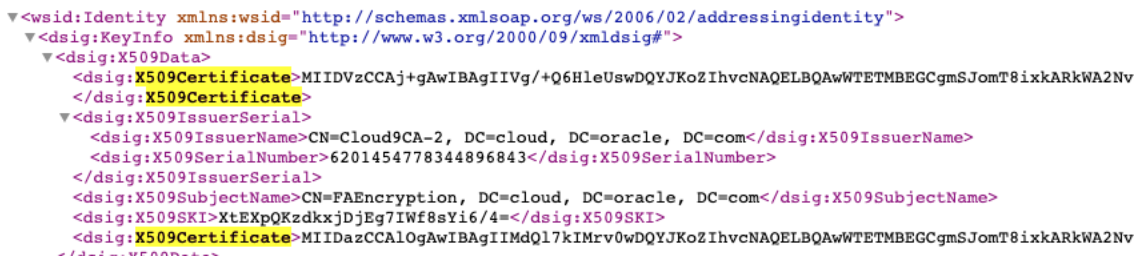
Option 1:

Oracle ERP cloud certificates can be retrieved from ERP Integration Service WSDL as follows:

In your browser, enter ERP Integration Service URL:

Eg. https://eeho.fa.us2.oraclecloud.com/fscmService/ErpIntegrationService?wsdl

In the WSDL, navigate to X509 certificates section as shown below:



Copy certificate string in a file from above as follows:

-----BEGIN CERTIFICATE-----

certificate string

-----END CERTIFICATE-----

Repeat above for second certificate. You will have two certificate files. For example, erp_cert1.cer and erp_cert2.cer.

Option 2:

Support need to open Collab SR to get Certificates:

Navigate to the Enterprise Manager:

Domain -> Security -> Keystore -> owsm-> keystore manage button

1. Obtain the cloudca certificate.
2. Obtain orakey sign.

Sample Callback Service WSDL

```
Sample Callback Service Wsdl
<?xml version="1.0" encoding="UTF-8" ?>
<wsdl:definitions
 name="ErpIntegrationCallbackservice"
 targetNamespace="http://xmlns.oracle.com/scheduler"
 xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
 xmlns:tns="http://xmlns.oracle.com/scheduler"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
```

```
 xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
 xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
 >
 <wsdl:types>
 <xs:schema version="1.0"
targetNamespace="http://xmlns.oracle.com/scheduler/types"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:complexType name="onJobCompletion">
 <xs:sequence>
 <xs:element name="jobDefName" type="xs:string"
minOccurs="0"/>
 <xs:element name="jobPackageName" type="xs:string"
minOccurs="0"/>
 <xs:element name="paramList" type="xs:string"/>
 </xs:sequence>
 </xs:complexType>
 <xs:simpleType name="state">
 <xs:restriction base="xs:string">
 <xs:enumeration value="WAIT"/>
 <xs:enumeration value="READY"/>
 <xs:enumeration value="RUNNING"/>
 <xs:enumeration value="COMPLETED"/>
 <xs:enumeration value="BLOCKED"/>
 <xs:enumeration value="HOLD"/>
 <xs:enumeration value="CANCELLING"/>
 <xs:enumeration value="EXPIRED"/>
 <xs:enumeration value="CANCELLED"/>
 <xs:enumeration value="ERROR"/>
 <xs:enumeration value="WARNING"/>
 <xs:enumeration value="SUCCEEDED"/>
 <xs:enumeration value="PAUSED"/>
 <xs:enumeration value="PENDING_VALIDATION"/>
 <xs:enumeration value="VALIDATION_FAILED"/>
 <xs:enumeration value="SCHEDULE_ENDED"/>
 <xs:enumeration value="FINISHED"/>
 <xs:enumeration value="ERROR_AUTO_RETRY"/>
 </xs:restriction>
 </xs:simpleType>
 </xs:schema>
 <xs:schema elementFormDefault="qualified" version="1.0"
targetNamespace="http://xmlns.oracle.com/scheduler"
 xmlns:ns1="http://xmlns.oracle.com/scheduler/types"
xmlns:tns="http://xmlns.oracle.com/scheduler"
 xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:import namespace="http://xmlns.oracle.com/scheduler/types"/>
 <xs:element name="onJobCompletion" nillable="true"
type="tns:onJobCompletion"/>
 <xs:complexType name="onJobCompletion">
 <xs:sequence>
 <xs:element name="requestId" type="xs:long"
form="unqualified"/>
 <xs:element name="state" type="ns1:state"
form="unqualified" minOccurs="0"/>
 <xs:element name="resultMessage" type="xs:string"
form="unqualified"/>
 </xs:sequence>
 </xs:complexType>
 </xs:schema>
 </wsdl:types>
 <wsdl:message name="onJobCompletionInput">
 <wsdl:part name="onJobCompletion" element="tns:onJobCompletion"/>
 </wsdl:message>
```

```
 <wsdl:portType name="ErpIntegrationCallbackService">
 <wsdl:operation name="onJobCompletion">
 <wsdl:input message="tns:onJobCompletionInput"
xmlns:ns1="http://www.w3.org/2006/05/addressing/wsdl"

ns1:Action="http://xmlns.oracle.com/scheduler/onJobCompletion"/>
 </wsdl:operation>
 </wsdl:portType>
 <wsdl:binding name="ErpIntegrationCallbackServiceSoapHttp"
type="tns:ErpIntegrationCallbackService">
 <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
 <wsdl:operation name="onJobCompletion">
 <soap:operation
soapAction="http://xmlns.oracle.com/scheduler/onJobCompletion"/>
 <wsdl:input>
 <soap:body use="literal" parts="onJobCompletion"/>
 </wsdl:input>
 </wsdl:operation>
 </wsdl:binding>
 <wsdl:service name="ErpIntegrationCallbackservice">
 <wsdl:port name="ErpIntegrationCallbackservicePort"
binding="tns:ErpIntegrationCallbackServiceSoapHttp">
 <soap:address
location="http://localhost:7101/finFunSharedErpIntegrationCallback/ErpIntegr
ationCallbackservice"/>
 </wsdl:port>
 </wsdl:service>
</wsdl:definitions>
```

Sample SAML assertion from Oracle ERP Cloud when invoking callback service.

```
<wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-
1.0.xsd"
env:mustUnderstand="1">
 <saml:Assertion xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion" MajorVersion="1" MinorVersion="1"
AssertionID="SAML-fCdgV0BqwQDU97xq6frjRw22" IssueInstant="2016-05-24T01:06:22Z"
Issuer="www.oracle.com">
 <saml:Conditions NotBefore="2016-05-24T01:06:22Z" NotOnOrAfter="2016-05-24T01:11:22Z"/>
 <saml:AuthenticationStatement AuthenticationInstant="2016-05-24T01:06:22Z"
AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:password">
 <saml:Subject>
 <saml:NameIdentifier
Format="urn:oasis:names:tc:SAML:1.1:nameidformat:unspecified">FINUSER1</saml:NameIdentifier>
 <saml:SubjectConfirmation>
 <saml:ConfirmationMethod>urn:oasis:names:tc:SAML:1.0:cm:bearer</saml:ConfirmationMethod>
 </saml:SubjectConfirmation>
 </saml:Subject>
 </saml:AuthenticationStatement>
 <dsig:Signature xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
 <dsig:SignedInfo>
 <dsig:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
 <dsig:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
 <dsig:Reference URI="#SAML-fCdgV0BqwQDU97xq6frjRw22">
 <dsig:Transforms>
 <dsig:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
 <dsig:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
 </dsig:Transforms>
 <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
 <dsig:DigestValue>IqAsJkAm490B8xQLk3+ztDPLggw=</dsig:DigestValue>
 </dsig:Reference>
```

```
 </dsig:SignedInfo>
 <dsig:SignatureValue>DqM35PHNh4RKvWiZ/QOBYieWl0a9sKk0eFYSISL2NRqK7gLNdH8mTUT3HlBl28
2b3FsJrEgK25cuV1f2suquyEzOeGUQTd71hueW4xbXg5bjUNAQ+RXjdP3TUr8wy9+Ftv1s2tobZSXyfpxmHDDcsgOE
P2MltkltjLKRyCDlHTrqUxBJDW+g5KvwgNhjadA/1ShlBeyA4uv2X3uxa/lyDGr5h82P6v+jUlXtnDjZW0lDphdjZOXx
+y943tYPx8SYDalaf20lb19lHl10nwm76nzo7gU/MAxS2d5uf1YsbH15NN2tD7amwpSznRTt8OgTwgru27XcFVQ+0F
G
jA4O26Q==</dsig:SignatureValue>
 <dsig:KeyInfo>
 <dsig:X509Data>
<dsig:X509Certificate>MIIDbDCCAlSgAwIBAgIGAVL8VT1fMA0GCSqGSIb3DQEBCwUAMHgxCzAJBgNVBAYT
AlVTMRAwDgYDVQQIEwdNeVN0YXRlMQ8wDQYDVQQHEwZNeVRvd24xFzAVBgNVBAoTDk15T3JnYW5pe
mF0aW9uMRkwFwYDVQQLExBGT1IgVEVTVElORyBPTkxZMRIwEAYDVQQDEwlDZXJ0R2VuQ0EwHhcNMTY
wMjIwMDEzOTUwWhcNMjEwMjE4MDEzOTUwWjBTMRMwEQYKCZImiZPyLGQBGRYDY29tMRYwFAYKCZI
miZPyLGQBGRYGb3JhY2xlMRIwEAYKCK8tzPmS9ielPXDmLnllaF+NvYf2YsQzurFxkzJ37PPeOplmr5bj83gyTKzsn
RLETN6j1cQUtCARa+QPuBYw9c2Y4gkoFSyKBS3nRG5ulJmFafoB3SBBfX93126V7rZBjZ8QrzsZhATTZQijs2a7s/X
2hNTuGhelWqyerlUrek5PErkPY7eYFijmwn5pcSokUV10Py6dxw+A==</dsig:X509Certificate>
 <dsig:X509IssuerSerial>
 <dsig:X509IssuerName>CN=CertGenCA, OU=FOR TESTING ONLY, O=MyOrganization, L=MyTown,
ST=MyState, C=US</dsig:X509IssuerName>
 <dsig:X509SerialNumber>1455932390751</dsig:X509SerialNumber>
 </dsig:X509IssuerSerial>
 <dsig:X509SubjectName>CN=service, DC=us, DC=oracle, DC=com</dsig:X509SubjectName>
 <dsig:X509SKI>kFgAwGeiKu8lliJe62UKOE2V0X8=</dsig:X509SKI>
 </dsig:X509Data>
 </dsig:KeyInfo>
 </dsig:Signature>
 </saml:Assertion>
```

Sample message that ERP Cloud returns to your callback web service.

Call back Response Payload

```
<?xml version="1.0" encoding="UTF-8"?>
<inputVariable>
	<part
		xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
		<ns0:onJobCompletion
			xmlns:ns0="http://xmlns.oracle.com/scheduler"
			xmlns:wsa="http://www.w3.org/2005/08/addressing">
			<requestId>17960</requestId>
			<state>SUCCEEDED</state>
			<resultMessage>{"JOBS":[{"JOBNAME":"Load Interface File for
Import","JOBPATH":"/oracle/apps/ess/financials/commonModules/shared/common/interfaceLoader","DOCUMEN
TNAME":"CCLFGL.CONCUR.CCLJEES.1031.zip","REQUESTID":"17957","STATUS":"SUCCEEDED","CHILD":[{"JOBNAM
E":"Transfer
File","JOBPATH":"/oracle/apps/ess/financials/commonModules/shared/common/interfaceLoader","REQUESTID":"
17958","STATUS":"SUCCEEDED"},{"JOBNAME":"Load File to
Interface","JOBPATH":"/oracle/apps/ess/financials/commonModules/shared/common/interfaceLoader","REQUEST
ID":"17959","STATUS":"SUCCEEDED"}]},{"JOBNAME":"Import
Journals","JOBPATH":"/oracle/apps/ess/financials/generalLedger/programs/common","REQUESTID":"17960","ST
ATUS":"SUCCEEDED"}],"SUMMARYSTATUS":"SUCCEEDED"}</resultMessage>
		</ns0:onJobCompletion>
	</part>
</inputVariable>
```

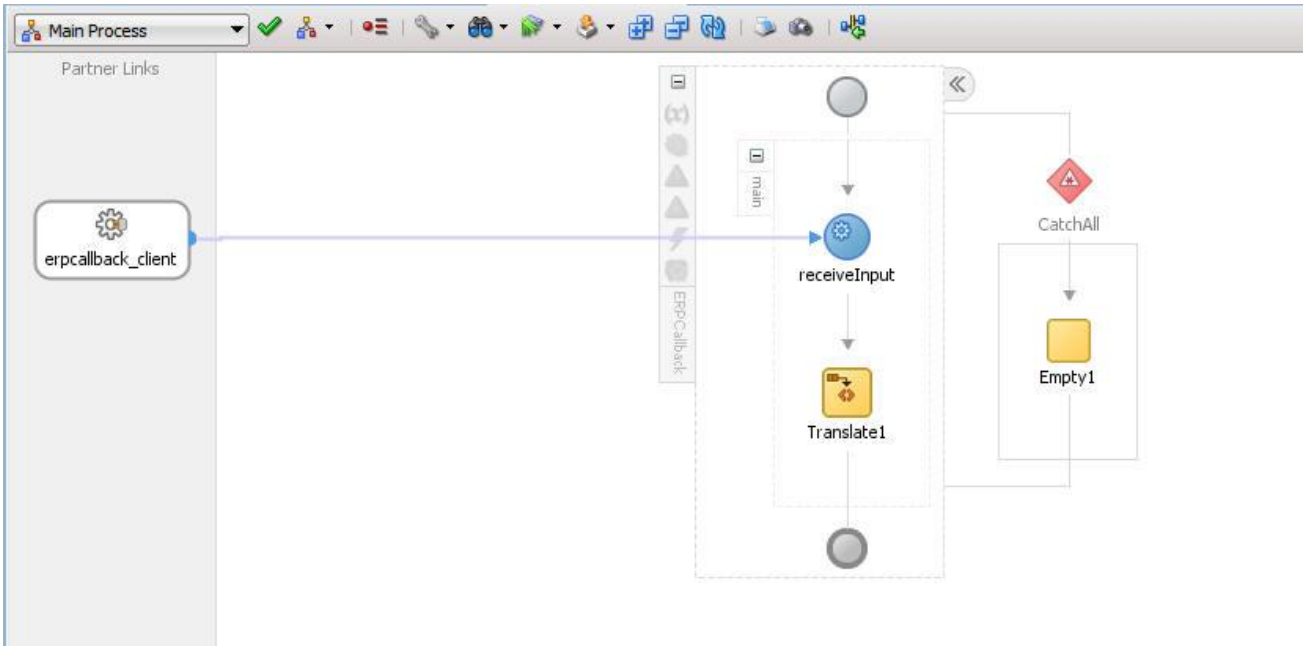
Implementation Steps

SOACS

Here, we are leveraging the Oracle Service Oriented Architecture Cloud Service (SOACS) deployed on Oracle's cloud Platform as a Service (PaaS) infrastructure.

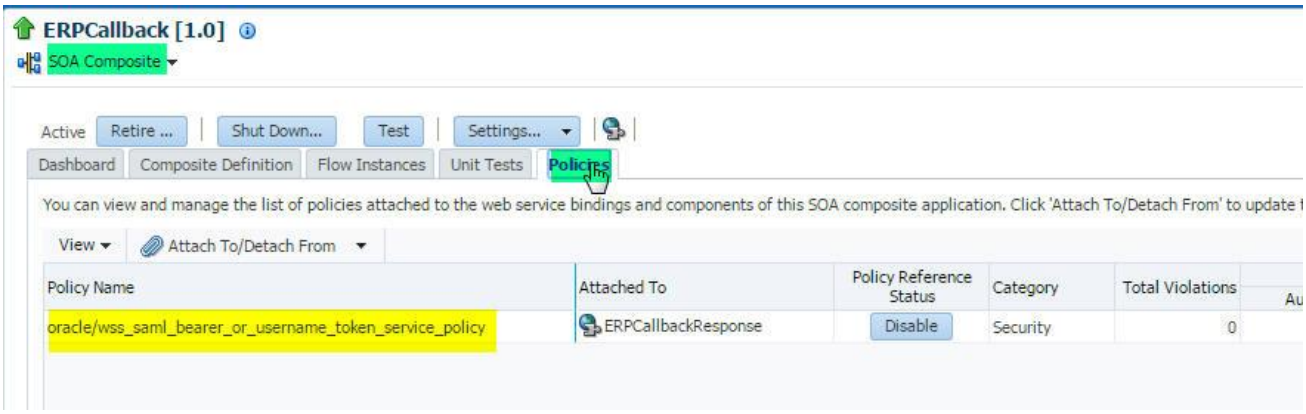
This is a typical SOACS composite:

This is a simple BPEL component to receive the callback.



Certificates must be in OWSM keystore stripe.

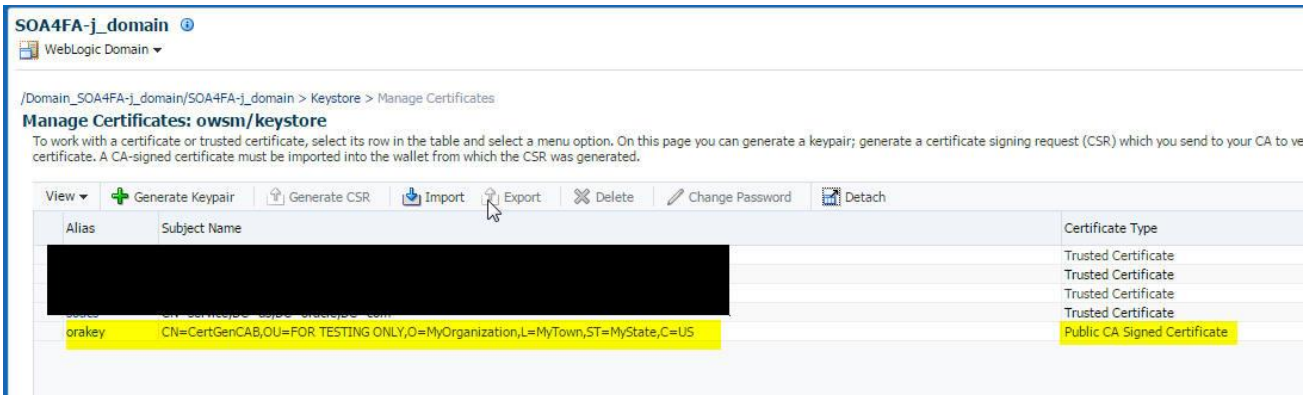
The following diagram illustrates the OWSM policy applied to your SOA Composite reference:



You must configure your SOACS instance as follows:

1.The SOACS instance must support issuer www.oracle.com (this is default in SOACS) and LDAP directory must have saml:NameIdentifier user name. In this sample it is FINUSER1.

2.The SAML assertion also contains digital signature that must be imported in SOACS's OWSM keystore. This certificate is in ERP integration WSDL. See this diagram:

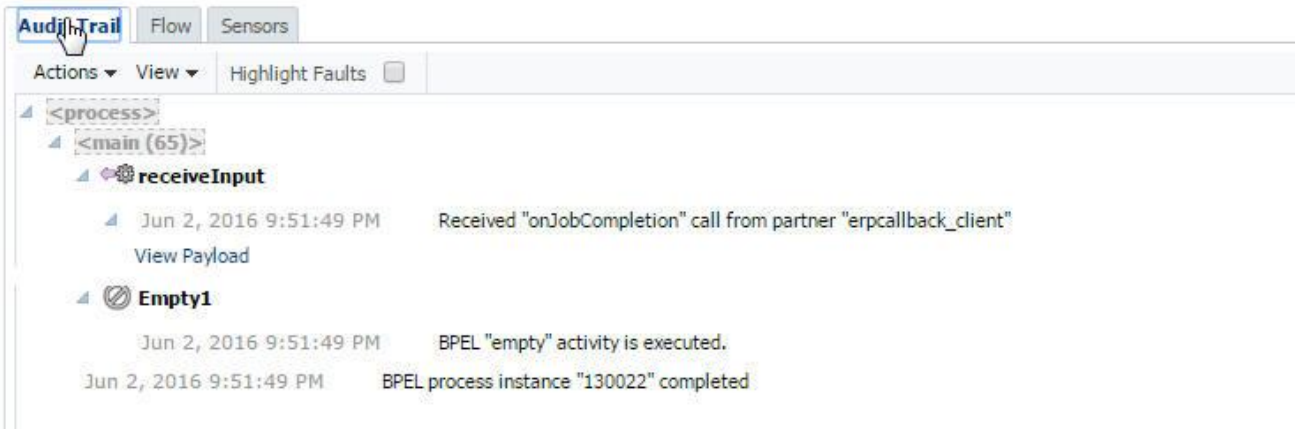


Execution and Run-Time Monitoring

Invoke ERP Integration Service Operation - importBulkData or exportBulkData.

In callbackURL, you must provide your callback web service endpoint URL that ERP Cloud will invoke upon job completion.

Following screen illustrates SOACS runtime instance and details:



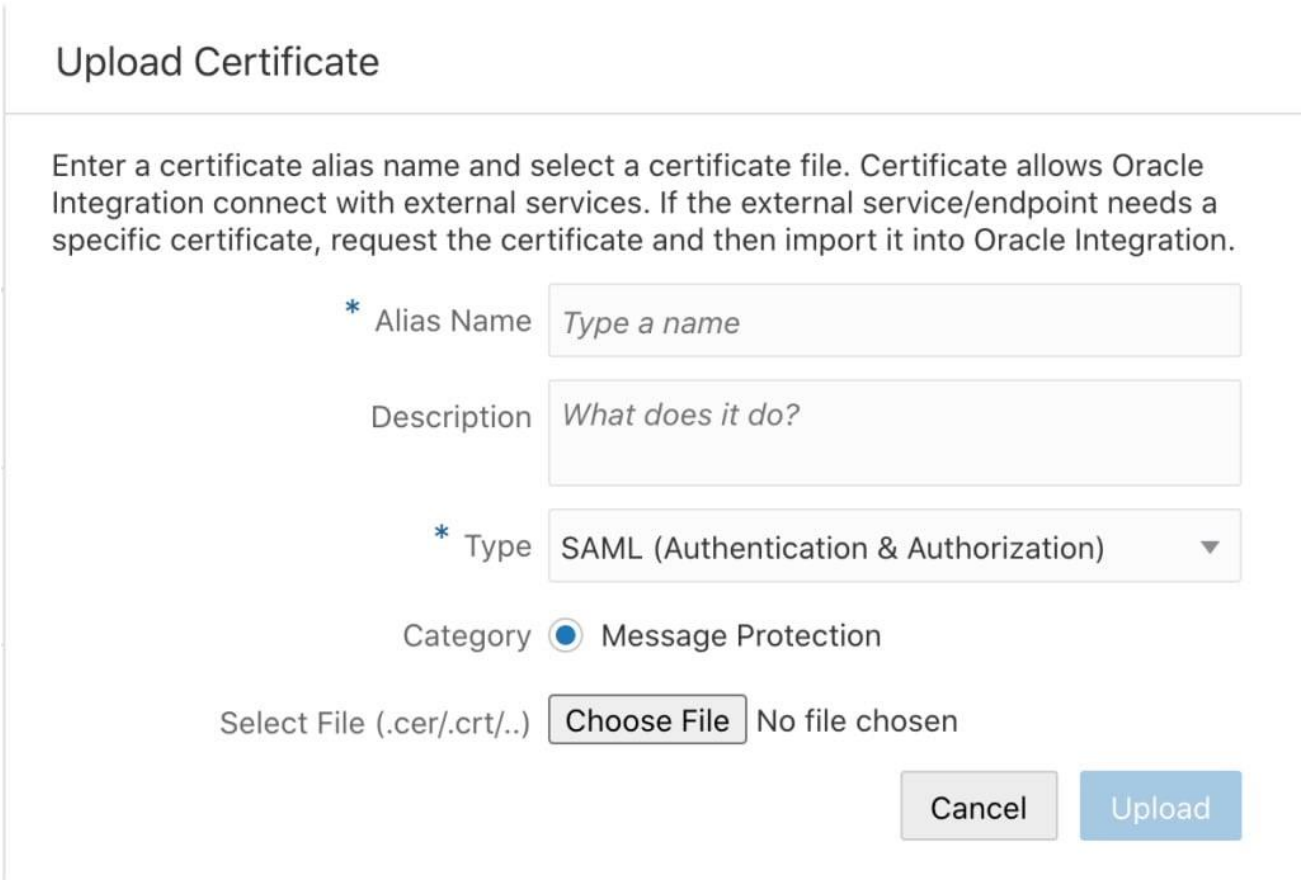
Oracle Integration Cloud (OIC)

Callback can be implemented based upon a service.

Steps to upload ERP Cloud Certificate in OIC

- Create a user like the Oracle ERP Cloud Adapter connection user in the My Services or Oracle Cloud Infrastructure Console. This user is linked to the Oracle Integration WebLogic security realm. The username must exactly match the Oracle ERP Cloud username. The password and email address can be anything. Ensure that this user has permissions to execute integrations in Oracle Integration. It is recommended that you have a role such as ServiceAdmin or ServiceDeveloper, which have executable permissions on integrations. Ensure that you select the Integration role for the Oracle ERP Cloud user in the Oracle ERP Cloud application.
- Import the certificates. From OIC home page, Click Settings > Certificates

  Click Upload to import certificates and select certificate type "SAML (Authentication and Authorization) and category "Message Protection" is automatically selected. Enter a unique alias name and select the certificate file.



Repeat the upload process for the second certificate.

For more information, see: Upload an SSL Certificate

Mulesoft

Uploading ERP Cloud Certificate in Mulesoft:

- The cloudCA and orakey_sign certificates need to be added to Mulesoft DLB (Dedicated Load Balancer) as client certificate. Refer to Mulesoft documentation link for steps to upload certificates.
- Please load the oracle client certificates in DLB as optional.
- As Mulesoft supports only one client certificate file(.pem) to uploaded on DLB, merge the cloudCA and orakey_sign into one file. The order is first put cloudca followed by orakey_sign. The certificate chain can be validated here.
- As Oracle client certificate is added as optional in DLB. Each application will need to check **X-SSL-Client Verify : None** header in all the application to determine if the requests need to be allowed or blocked.
- In Application - no need to include Oracle certs in the trust store.

Uploading Customer's Certificate in Mulesoft:

In Mulesoft DLB if additional customer CA certificate need to be loaded in the public key, Intermediate certificate must be loaded along with root as part of public key.

Run Wget to confirm if Call back service is reachable from Oracle FA Servers:

wget -e use_proxy=yes -e https_proxy=proxy-ip:80 https://host:port/dummyCallbackServiceURL?wsdl

Appendix 11: Creating a Job Property File for the importBulkData Operation

This appendix focuses on how to generate a property (or manifest) file that requires the job package and name including list of job parameters when submitting respective bulk import job. In Oracle Integration Cloud (OIC), use the ERP Adapter (out-of-the-box simplified) feature to implement bulk import that automates the uploading of data/property zip file to Content Manager (UCM) followed by invoking the ERP Integration service.

Import Job Property File Format

The property file submitted with importBulkData operation includes the job definition and package names as well as the job parameters of the object being imported. You must generate and add the Job Properties File as part of the ZIP file.

The job property file is in the comma separated format with the following content in specified order:

<job package name>,<job definition name>, <ZIP file prefix>,<Param1>,....<ParamN>

The name could be anything, but the extension must be ".properties".

See "*Appendix 18: How to get job definition & package name including parameters*" to derive the package and definition name of the import job. The third column (ZIP File Prefix) is the name of the zip file that contains the data file. The fourth to N columns are lists of parameters that are required to load and import the data into ERP Cloud.

Example:

Your zip file name is GL_BU_ABC_12345678.zip and contains GlInterface.csv and jobDetails.properties. The value of the third column in jobDetails.properties could be GL_BU_ABC.

Final content in .properties file

/oracle/apps/ess/financials/generalLedger/programs/common/,JournalImportLauncher,GL_BU_ABC,<Data Access Set>,<source>,<Ledger>,<Group ID>,<Post Account Errors to Suspense>,<Create Summary Journals>,<Import Decriptive Flexfields>

Example: /oracle/apps/ess/financials/generalLedger/programs/common/,JournalImportLauncher,GL_BU_ABC, 1061,Balance Transfer,1,ALL,Y,Y,N

Reusing the same Job property file for similar imports

- You can reuse the same job property file for similar imports by uploading the file to the respective UCM account from the Oracle Fusion Applications UI or using the uploadFileToUcm operation.
- To reuse the parameter file, you must provide the file name in the request payload as follows:

<ns1:jobOptions> ***JobDetailFileName****=<fileName.properties></ns1:jobOptions>*

Deciding factors to provide Job details

Below factors will support the decision for choosing between alternative approach for providing job details:

- When job details (Job Package, Job Name and Parameter) are always constant for any specific integration, provide the job details in Web Service Payload.
- When job details are dynamic and frequency of modifications is significantly low, use the same property file for similar imports.
- When job details are dynamic most of the time, add property file inside the ZIP file.

Appendix 12: Manual Inbound (Import) Steps

Transferring Data Files to Oracle WebCenter Content Server

After you generate the ZIP file that contains the CSV data import file, transfer the ZIP file to the Oracle WebCenter Content Server.

Use any of the following methods to transfer the file:

- File Import and Export page in Oracle Fusion Applications: Manual flow
- Oracle WebCenter Content Server home page

File Import and Export

Use the File Import and Export page to access the WebCenter Content Repository. For example, each Oracle ERP Cloud instance connects to a single Oracle WebCenter Content server for content management.

For more information about the use and administration of content management:

- Oracle WebCenter Content Server User's Guide
- Oracle WebCenter Content Server System Administrator's Guide

References for Using Content Management

The File-Based Data Import guides on the Oracle Help Center (http://docs.oracle.com) provide the objects to upload and download, including templates for external data integration. For general access to content management, including the metadata and manage accounts, use the Oracle WebCenter Content Server's standard service user interface.

The following table provides additional resources for more information:

Topic	Resource
Content server	Oracle WebCenter Content User's Guide for Content Server
Creating WebCenter content accounts	Oracle WebCenter Content System Administrator's Guide for Content Server
Naming accounts used in import and export processes	Files for Import and Export section

Programmatic upload and download to content management	Oracle WebCenter Content System Administrator's Guide for Content Server
Roles	Oracle Fusion Applications Common Security Reference Manual

Managing Files for Import and Export

Import data or export data out of Oracle ERP Cloud using content and processes repositories. Integration specialists stage data for import and export. Application administrators run processes to import data in the content repositories to the application transaction tables or retrieve data exported from the applications.

Managing files for import and export involve the following:

Using File Import and Export page to upload or download file

The File Import and Export page enables you to upload or download content from the document repository of Oracle WebCenter Content Management. The search criteria section is limited to the minimum metadata of content management records needed for file import and export.

To access the File Import and Export page:

1. From the Navigator, click **Tools**.
2. Click **File Import and Export**.

The maximum file size using the File Import and Export page is 2 GB.

Interacting with Content Management

When you use the File Import and Export page, you are assigned one or more accounts in the content management server. These accounts organize and secure access to the content items.

Interaction between the File Import and Export page and Oracle WebCenter Content server requires securing content in an account. Oracle provides predefined accounts in the Oracle WebCenter Content server.

File import and export include the following concepts:

- Security
- Searching records
- Accessing content in a new account
- Account names
- Deleting files
- Uploading for import
- Downloading for export
- File size

Security

Use the File Import and Export Management duty role to access the File Import and Export page. This duty role is included in the predefined role hierarchy for integration specialist roles and product family administrator roles.

The files in Oracle WebCenter Content Management server are associated with an account. The users who have the access privileges to a specific account can work with content items that belong to that account.

Note

You can only upload and download files to and from the content management server that is associated with the accounts that you are entitled to access.

The Oracle WebCenter Content Management server does not support trailing slashes (/) as part of the account name. Account names are appended with a dollar sign ($) to ensure each account is unique. Account names are dynamic so that if they overlap (one name is completely contained in another, longer name, such as the US and US Sales), each account is considered as discrete by access grants. Security such as virus scanning is handled by the underlying integrated content management.

Searching Records

A record in the Oracle WebCenter Content Management server contains metadata used to access the file. When you run the 'Load Interface File for Import' process for a file the file meta-deta is updated with comment 'ProcessedBy=<LOAD_REQUEST_ID> '

Accessing Content in a New Account

When you create a new account in the Oracle WebCenter Content Management server and the content server is not restarted, the access to the content in the new account from the File Import and Export page is delayed until the policy store is updated.

Account Names

If you create custom accounts for importing or exporting data, use the following naming conventions for the account:

- To avoid partial string matching, do not include a slash (/) at the beginning or use a dollar sign ($) at the end.
- Use a dollar sign and slash ($/) as a separator in the hierarchical structure.

The File Import and Export page transforms account names by removing the dollar signs ($). For example, fin$/journal$/import$ displays as fin/journal/import.

The Remote Intradoc Client (RIDC) HTTP Command-Line Interface (CLI) transforms the account name you specify without the dollar signs ($) to one that includes them. For example, fin/journal/import becomes fin$/journal$/import$ in the Oracle WebCenter Content Management server.

You must transfer files to these predefined accounts in content management that correspond to the interface table or assets of interest. For a list of the target accounts in each interface table, see *Appendix 3: Predefined Target UCM Accounts.*

Deleting Files

You can delete one file at a time in the File Import and Export page. To delete multiple files simultaneously from the content repository, use the standard service page in the Oracle WebCenter Content Management server.

Uploading for Import

To create a record, you must specify an account and the file. The account you specify determines which import process is used.

You can upload any file formats that can be parsed by the content repository, such as any MIME or content types. However, the format uploaded must conform to the requirements of the import process, such as a CSV file for the Load Interface File for Import process.

Downloading for Export

The export data processes create files in the content management server. Records in the search results table of the File Import and Export page provide links to the files for download.

Load Interface File for Import Process

The Load Interface File for Import process loads external setup or transaction data from the data file on the content management server to the relevant product interface table(s). You can run this process from the Scheduled Processes page on a recurring basis as needed.

Before running this process, you must:

1. Prepare your data file.
2. Transfer the data file to the content management server.

The following table describes the parameters for this process:

Parameter	Description
Import Process	Select the target import process.
Data File	Choose the data file from the choice list.

Importing Data

The destination for your external data is the product application data table(s) of your Oracle ERP Cloud application.

Importing data into the application tables involves the following:

- Loading data into the product interface table(s)
- Finding and submitting the applicable import process

Loading Data into Interface Tables

Interface tables are intermediary tables that store your data temporarily while the application validates format and structure. Run the Load Interface File for Import scheduled process to load data from the data file into the interface table that corresponds to the template that you use to prepare the data.

To load your data into interface tables, submit the Load Interface File for Import scheduled process using the following steps:

1. From the Navigator, click **Tools**.
2. Click **Scheduled Processes**.
3. Click **Schedule New Process.**
4. Search and select the Load Interface File for Import job.
5. On the Process Details page:
    a. Select the target import process.
    b. Enter the data file name.
6. Submit the process.

If the process is successful, the status is SUCCEEDED and the process populates the interface tables. If the process isn't successful, the status is ERROR. For more information on correcting load errors, see the Correcting Load Process Errors.

The Load Interface File for Import process ends in error when the load of the data file fails for any individual row. The Load File to Interface child process ends as an error or warning. All rows that were loaded by the process are deleted and the entire batch of records is rejected.

Correcting Interface Data Errors

Correcting errors:

1. Review the upload error logs.
2. Change any structural or formatting anomalies in the data.
3. Generate a ZIP file containing the CSV files using the respective import template.
4. Upload the corrected file to the UCM server and resubmit the Load Interface File for Import process.

Repeat these steps until the process successfully loads all the data.

Correcting Import Process Errors

If the import process fails with errors:

1. Review the errors in the import log.
2. Correct the error records using the applicable ADFdi correction spreadsheets.

For a list of import processes and their corresponding ADFdi correction spreadsheets, see *Appendix 7: Error Handling for Import Jobs.*

If auto purge is enabled in your import process, then you cannot use ADFdi. Use these steps:

1. Download the purge erroneous ZIP file from the File Import and Export page.
2. Select the erroneous data records from the interface file and correct them.

Follow the FBDI process to resubmit the corrected data.

Appendix 13: Managing PGP Encryption Keys

Managing PGP Certificates

Certificates establish keys for the encryption and decryption of data that Oracle Cloud applications exchange with other applications. The Oracle Fusion Applications Security Console is an easy-to-use administrative interface that you access by selecting **Tools** → **Security Console** on the home page or from the Navigator. Use the Certificates page in the Security Console functional area to work with PGP certificates.

A PGP certificate consists of a public key and a private key. The Certificates page displays one record for each certificate. Each record reports these values:

- Type: For a PGP certificate, **Public Key** is the only type.
- Private Key: A check mark indicates that the certificate's private key is present. For either certificate format, the private key is present for your own certificates (those you generate in the Security Console). However, the private key is absent when a certificate belongs to an external source and you import it using the Security Console.
- Status: For a PGP certificate, the only value is **Not Applicable**. (A PGP certificate has no status.)
- For each certificate, click the button at the end of the row to display a menu of actions appropriate for the certificate. Alternatively, to view details for a certificate, select its name ("alias"). Actions include:
    - Generating certificates
    - Importing and exporting PGP certificates

- Deleting certificates

Generating Certificates

For a PGP certificate, one operation creates both the public and private keys. From the Certificates page, select the **Generate** option. On the Generate page, select the certificate format **PGP**, and enter values appropriate for the format.

For a PGP certificate, these values include:

- An alias (name) and passphrase to identify the certificate uniquely.
- The algorithm by which the keys are generated, DSA or RSA.
- A key length, select **1024**.

Figure 49: Security Console Certificates main page

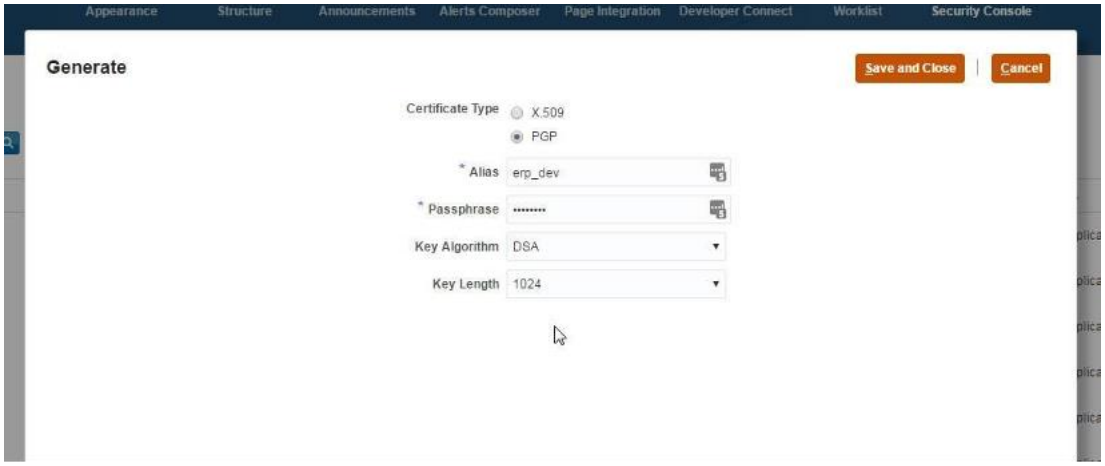


Figure 50: PGP Generate Certificates dialog window.

Importing and Exporting PGP Certificates

For a PGP certificate, you export the public and private keys in separate operations. You can import only public keys. The assumption is that you will import keys from external sources, which will not provide their private keys to you.

To export an Oracle Fusion public key:

1. From the Certificates page, select the menu in the row of the certificate that you want to export. Alternatively, open the details page for that certificate and select its **Actions.**
2. In either menu, select **Export**, then **Public Key** or **Private Key**.
3. If you select **Private Key**, provide its passphrase. (The public key does not require one.)
4. Select a location for the export file. By default, this file is called [alias]_pub.asc or [alias]_priv.asc

To import a Customer's PGP public key:

1. On the Certificates page, click **Import.**
2. On the Import page, select **PGP** and specify an alias (which does not need to match the alias of the file you are importing).
3. Browse for the public key file, and then select **Import and Close**.

The Certificates page displays a record for the imported certificate, with the **Private Key** cell unchecked.

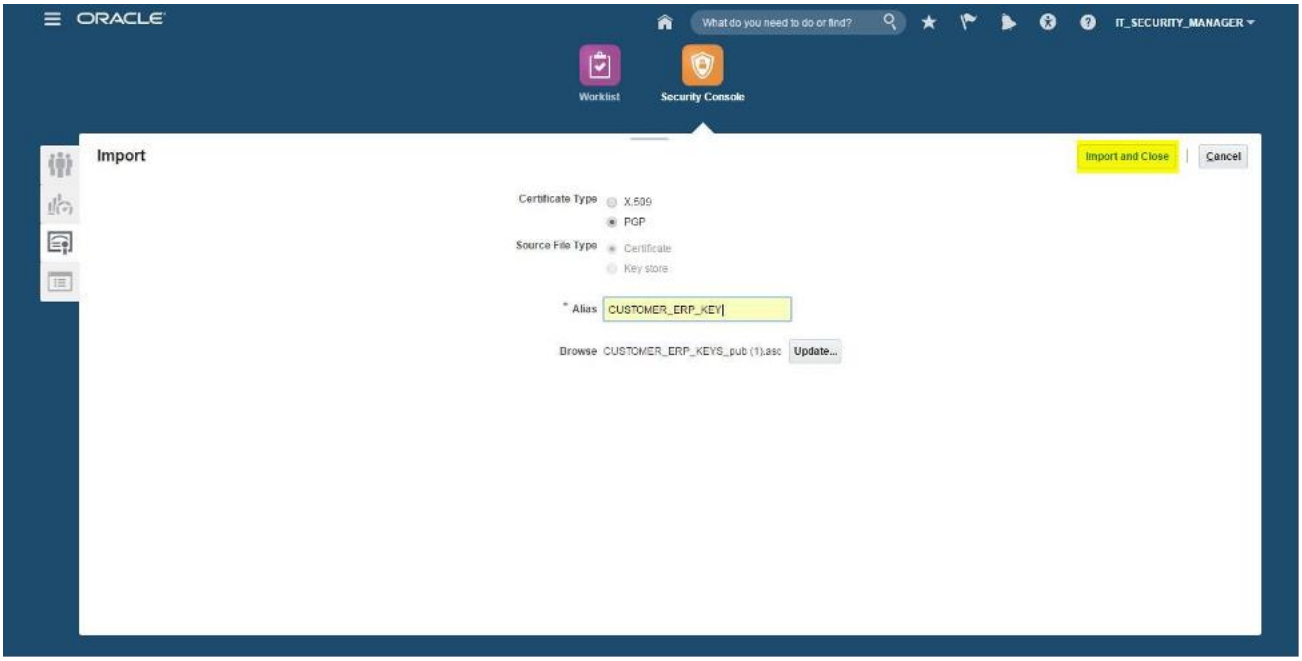


Figure 51: PGP Import Certificate page

Deleting Certificates

You can delete PGP certificates.

In the Certificates page, select the menu in the row of the certificate that you want to delete. Alternatively, select the **Actions** menu on the Details page for that certificate. In either menu, select **Delete**, then review and accept the resulting warning message as appropriate.

Appendix 14: How to Encrypt and Decrypt a Data File

Encrypt an Inbound Data File from your Linux On-Premise System

In Linux, you can use "gpg" to encrypt a data file. After PGP keys are generated, you must import the Oracle ERP Cloud public key as follows:

```
gpg --import <MY_ERP_KEY_pub.asc>

###Verify the imported key using this command gpg --list-keys
```

Figure 52: Sample command to import PGP public key

Once the public key is imported, use the following command to encrypt your inbound data file:

```
gpg --cipher-algo=AES -r=<alias> --encrypt <my_data_file>.zip
```

Figure 53: Sample command to encrypt ZIP file

The encrypted file will be renamed as <my_data_file>.zip.gpg.

Decrypt an Outbound Oracle ERP Cloud Data File in your Linux On-Premise System

To decrypt an outbound Oracle ERP Cloud data file, you must first import a customer's private key as follows:

```
gpg --allow-secret-key-import --import <my_private.asc>

###Verify the imported key using this command gpg --list-keys
```

Figure 54: Sample command to import PGP private key

Once a customer's private key is imported, use the following command to decrypt your outbound data file:

```
gpg --decrypt <EncryptedFileName> > <DecryptedFileName>
```

Figure 55: Sample command to decrypt file

Appendix 15: Large File Optimization (MTOM) Proxy Client Code Changes

The following example illustrates sample Java code changes for MTOM attachment:

uploadToFileUcm

```
//Example for uploadToFileUcm ¿ similar process for importBulkData

/* Extend your code for MTOM and the <content> will be generated as follows:
```

```
<Content
xmlns="http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationService/"
xmlns:xmime="http://www.w3.org/2005/05/xmlmime" xmime:contentType="application/octet-
stream"<xop:Include xmlns:xop="http://www.w3.org/2004/08/xop/include" href="cid:6098af03-52ff-4a40-9665-
c89085ac2f3a"></Content>

*/

private static BinaryElementPathInfo prepareAttachmentInfo() {


   Map < String, List < String >> xpaths = new HashMap < String, List < String >> ();
   List < String > paths = new ArrayList < String > ();
   paths.add("//typ:uploadFileToUcm/typ:Document/erp:Content");
   paths.add("//typ:uploadFileToUcm/typ:Document/erp:Content/xop:Include");
   xpaths.put("GenericRequest", paths);


   Map < String, String > prefixes = new HashMap < String, String > ();
   prefixes.put("typ", TYP_NAMESPACE);
   prefixes.put("erp", ERP_NAMESPACE);
   prefixes.put("xop", XOP_NAMESPACE);


   BinaryElementPathInfo elemInfo = new BinaryElementPathInfo(xpaths, prefixes);
   return elemInfo;


}

private static String ERP_NAMESPACE =
"http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationService/";
private static String XOP_NAMESPACE = "http://www.w3.org/2004/08/xop/include";
private static String TYP_NAMESPACE =
"http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationService/types/";
public static void setupMtomAttachments(SOAPMessage requestMessage, BinaryElementPathInfo
binaryElementPathInfo

                                    ) throws SOAPException {

      if (requestMessage == null)

                                    return;

      if (binaryElementPathInfo == null || binaryElementPathInfo.getXpathExpressions() == null
||              binaryElementPathInfo.getXpathExpressions().isEmpty())

                                    return;

      SOAPBody body = requestMessage.getSOAPBody();
```

Sample code snippet to enable MTOM

```
List < Element > binElemList = getAllBinaryElements(body, binaryElementPathInfo);
if (binElemList == null)

                        return;

for (Element theElm: binElemList) {
```

```
final OracleSOAPElement theSOAPElem;

if (theElm instanceof OracleSOAPElement)
                                        theSOAPElem = (OracleSOAPElement) theElm;

else {

  WsMetaFactory jrfMetaFactory = WsMetaFactory.newInstance(ImplType.JRF);
  SOAPFactory fact = jrfMetaFactory.createSOAPFactory();
  theSOAPElem = (OracleSOAPElement) fact.createElement(theElm);
  theElm.getParentNode().replaceChild(theSOAPElem, theElm);

}

theSOAPElem.setDataHandler(new DataHandler(new DataSource()  {
  public InputStream getInputStream() throws IOException  {

    {

      java.io.InputStream is = new FileInputStream(new File("C:\\temp\\test.zip"));

      return is;

    }

  }

  public OutputStream getOutputStream() throws IOException {

    throw new IOException();

  }

  public String getContentType() {
    return "application/octet-stream";

  }

  public String getName() {
    return theSOAPElem.getLocalName();

  }

}));
```

```
} //for every binary element   requestMessage.setProperty(MessageImpl.PACKAGING_STYLE,
MessageImpl.MTOM);   requestMessage.setProperty(MessageImpl.ATTACHMENT_STYLE_PACKAGING,"true");   requ
estMessage.setProperty(OracleSOAPMessage.MTOM_THRESHOLD, 1024);


}
```

Appendix 16: Purge - UI Based Approach

In addition to the automated approach, interface and error data can also be purged through the Oracle ERP Cloud user interface. These are the steps to execute the purge process through the user interface:

1. From the Oracle ERP Cloud home page, select Navigator **→Tools → Scheduled Processes** .
2. Click **Schedule New Process.**
3. Search for and select the **Purge Interface Tables** process name.
4. Enter the data as defined in the table below.

Parameter Name	**Description**	**Mandatory**
Purge Process Intent	There are three options: 1. File-based data import – Purge FBDI data. 2. Maintenance – Extend the number of days for the auto purge process to extract data and upload to UCM. The default setting is 30 days. 3. Other - Purge Non-FBDI data.	Yes
Import Process	Select the applicable import process name.	Yes
Load Request ID	Enter the Load Request ID or enable an ID range.	Yes
Extract Data	Yes/No. The default value is No. If the value is Yes, the process will extract all the data being selected for purge and upload the data to UCM for future audit reference.	No

Purge FBDI Object Data using a Single Load Request ID

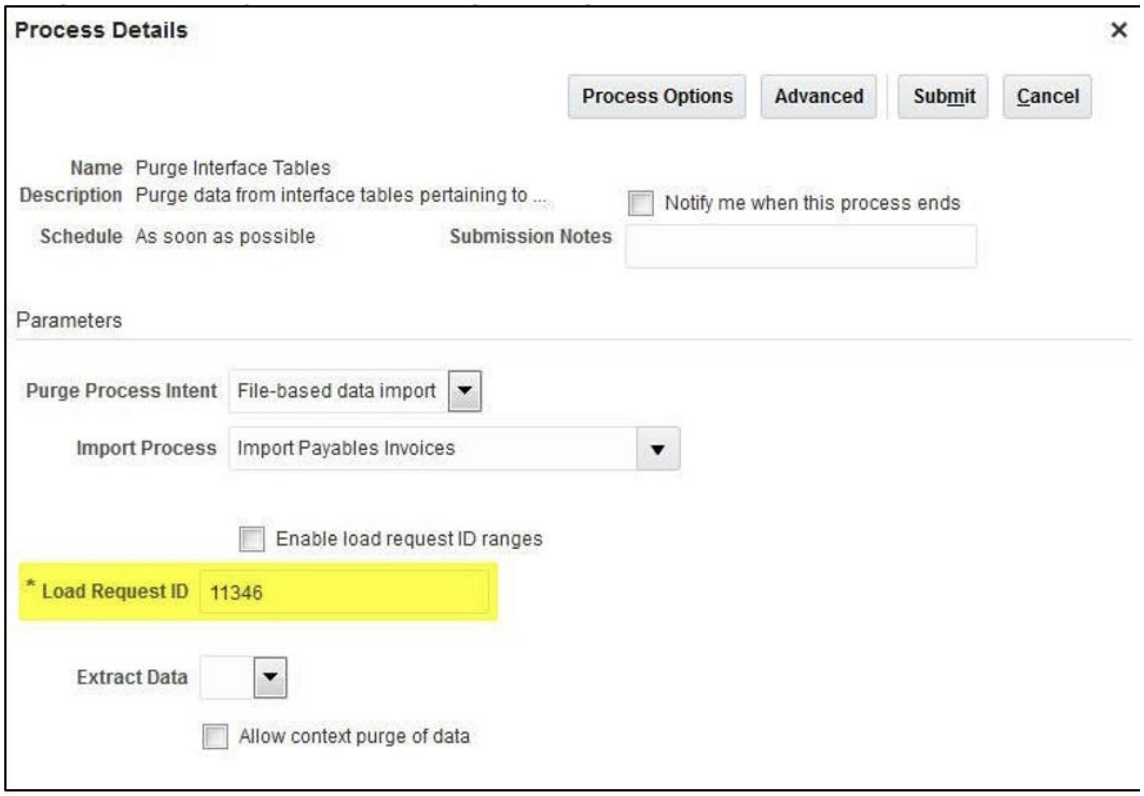


Figure 56: Sample of a purge using load request ID

Purge FBDI Object Data using a Range of Load Request IDs

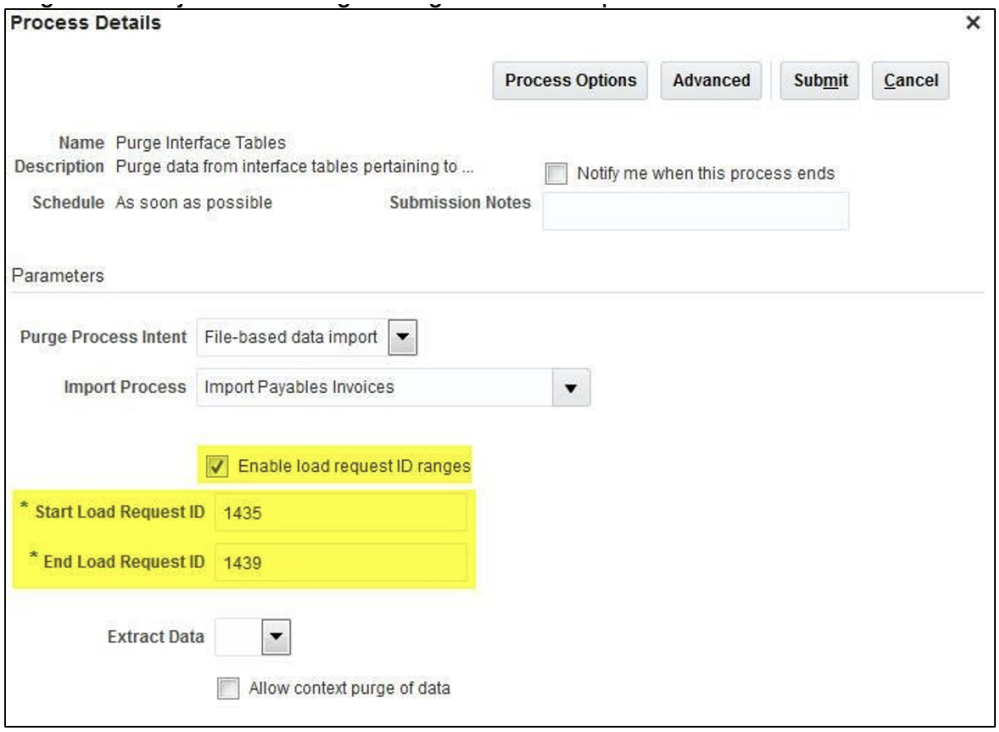


Figure 57: Sample of a purge using a range of load request IDs

Caution: Load Request ID range should be chosen wisely, keep that as narrow as possible so that there are no performance issues. For purging multiple load Request ID data at once, use 'Maintenance' intent.

Purging Non-FBDI Data

Non-FBDI data can be purged by selecting the Purge Process Intent as **Other** and providing the Import Request ID.

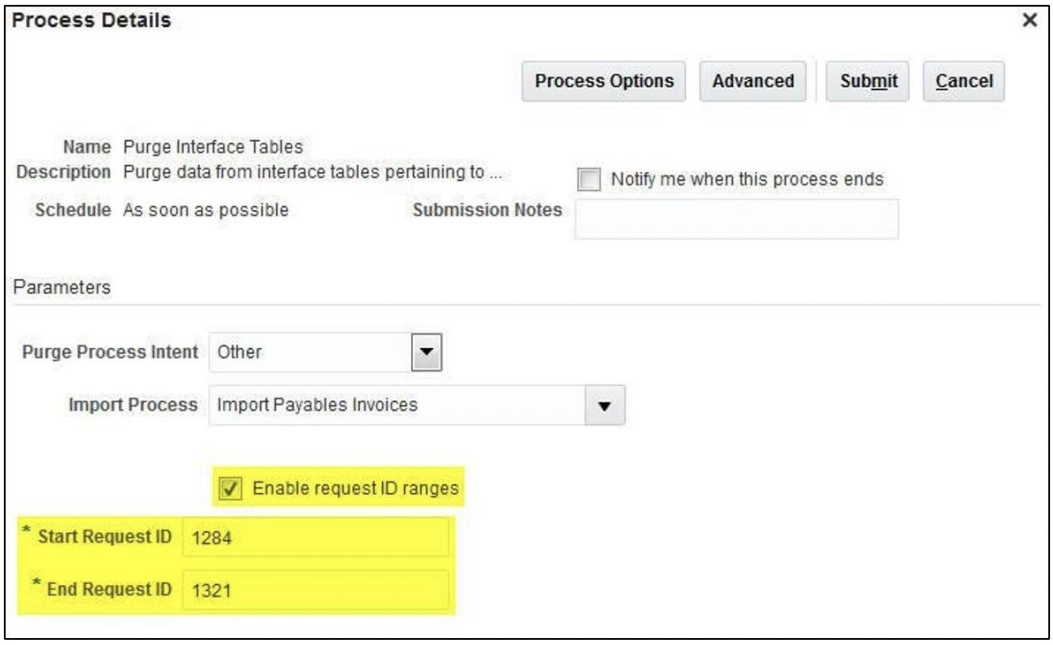


Figure 58: Sample of purging non-FBDI data

Caution: Request ID range should be chosen wisely, keep that as narrow as possible so that there are no performance issues. For purging multiple load Request ID data at once, utilise 'Maintenance' intent.

Purging Data in Maintenance mode

Both FBDI and Non-FBDI data older the provided age can be purged with maintenance intent. Non-FBDI purge is supported as per registered predicate.

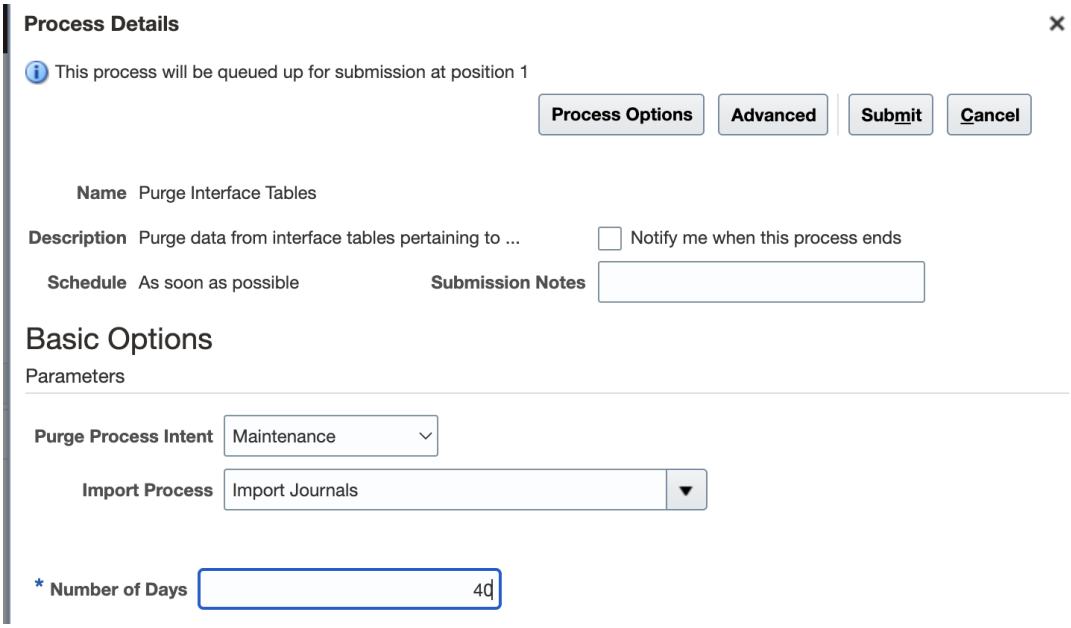


Figure 59 : Sample of purging in maintenance mode

Caution: The number of days value should always be less than ESS request history retention period (typically 60 days) and greater than 30 days.

ORACLE

Appendix 17: Steps to Get list of all Supported Import Processes for Load Interface File for Import Job

To get the details of all import processes supported by the **Load Interface File for Import** job, use the **getInboundProcesses** operation of **ERPIcsIntegrationService**. The details include InterfaceId, Import Process Display Name, UCM account, Import Job Path, Import Job Name, Job Description.

URL: https://<host:port>/fscmService/ErpIcsIntegrationService?WSDL

Example SOAP Request Payload

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:typ="http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationService/types
/">
  <soapenv:Header/>
  <soapenv:Body>
    <typ:getInboundProcesses/>
  </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response Payload

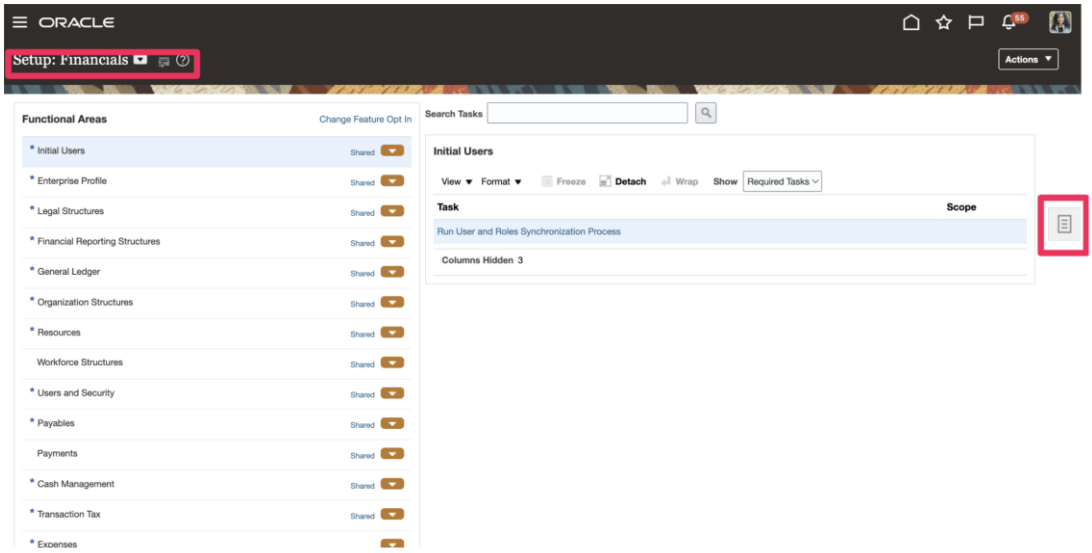
```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsa="http://www.w3.org/2005/08/addressing">
  <env:Header>
    <wsa:To>http://www.w3.org/2005/08/addressing/anonymous</wsa:To>

<wsa:Action>http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationService//Erp
IcsIntegrationService/getInboundProcessesResponse</wsa:Action>
    <wsa:MessageID>urn:uuid:ece2e950-8516-4be1-98bc-049879852284</wsa:MessageID>
    <wsa:RelatesTo>urn:uuid:caabe942-ccd2-4790-96f8-2d1317b05f9b</wsa:RelatesTo>
  </env:Header>
  <env:Body>
    <ns0:getInboundProcessesResponse
xmlns:ns0="http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationService/types
/">
      <result
xmlns="http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationService/types/">[
{"InterfaceId":"1","JobDisplayName":"Import Payables
Invoices","UcmAccount":"fin/payables/import","ImportJobName":"/oracle/apps/ess/financials/payables/invoices/tr
ansactions;APXIIMPT","JobDescription":"Creates Oracle Fusion Payables invoices from invoice data in the open
interface tables."},{"InterfaceId":"120","JobDisplayName":"Import Payables Payment
Requests","UcmAccount":"fin/payables/import","ImportJobName":"/oracle/apps/ess/financials/payables/invoices/t
ransactions;APXPRIMPT","JobDescription":"Imports the payment request for One Time Payments. The process
creates the party, party site, payee, payee bank account, and invoices from data in the Payment Request Interface
table."}....]</result>
    </ns0:getInboundProcessesResponse>
  </env:Body>
</env:Envelope>
```

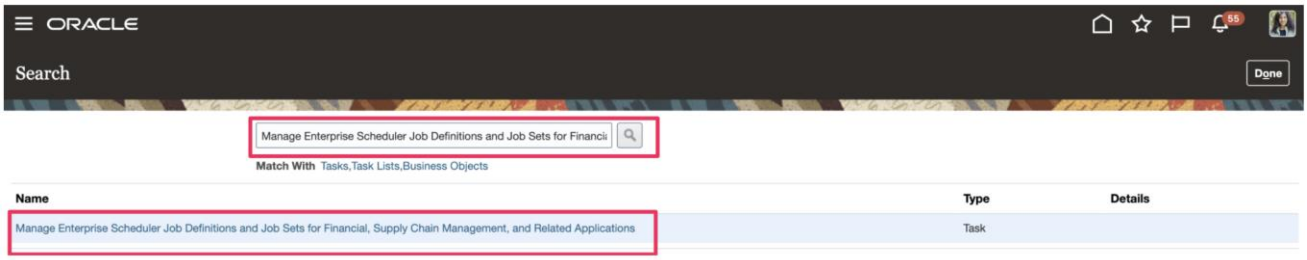
Alternatively, refer to Doc ID 2863486.1 to get a list that includes **Financials** Import Processes supported by the **Load Interface File for Import** job along with frequently used Import Processes from Projects and Procurement family.

Appendix 18: How to Get Job Definition & Package Name Including Parameters

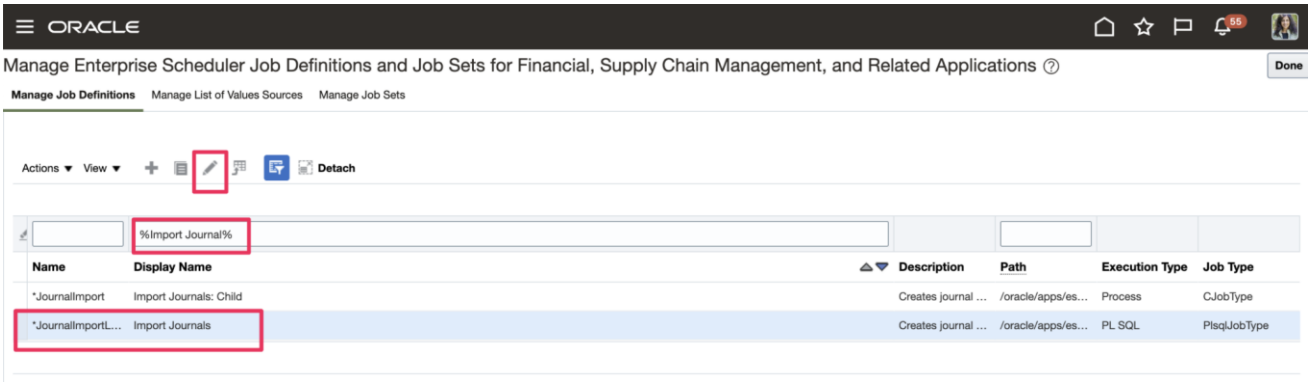
1. Open Fusion Applications main page.
2. Click the **Navigator**, and then click **Setup and Maintenance**.
3. On the Setup page, select your setup offering such as **Financials.**



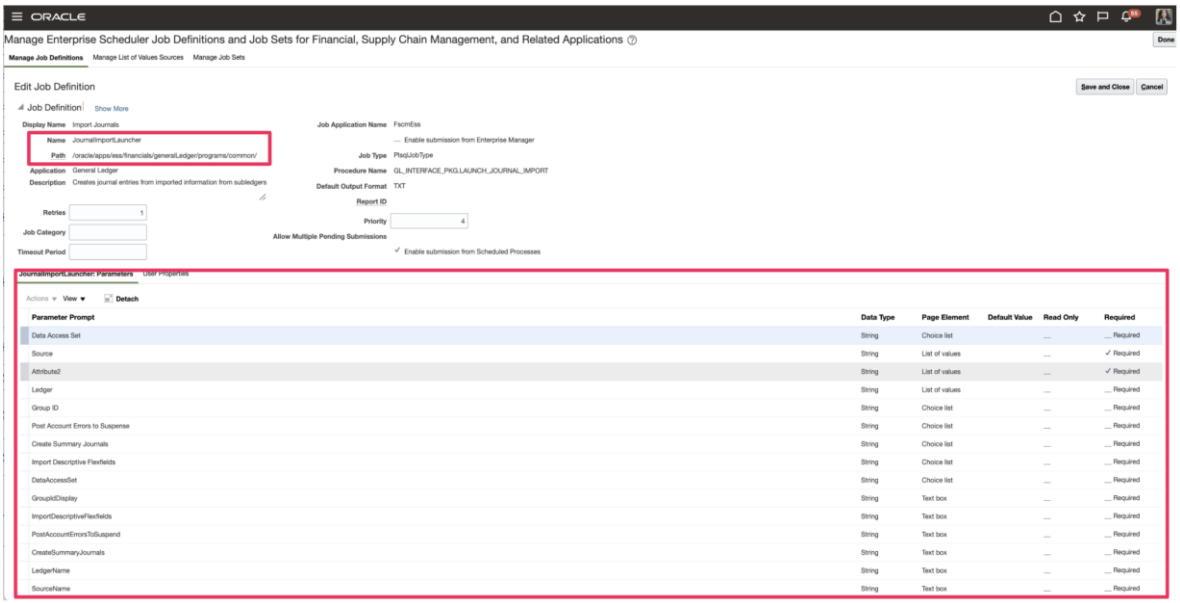
4. Click on the **Tasks** icon as shown above and search for "Manage Enterprise Scheduler Job Definitions and Job Sets for Financial, Supply Chain Management, and Related Applications**"**.



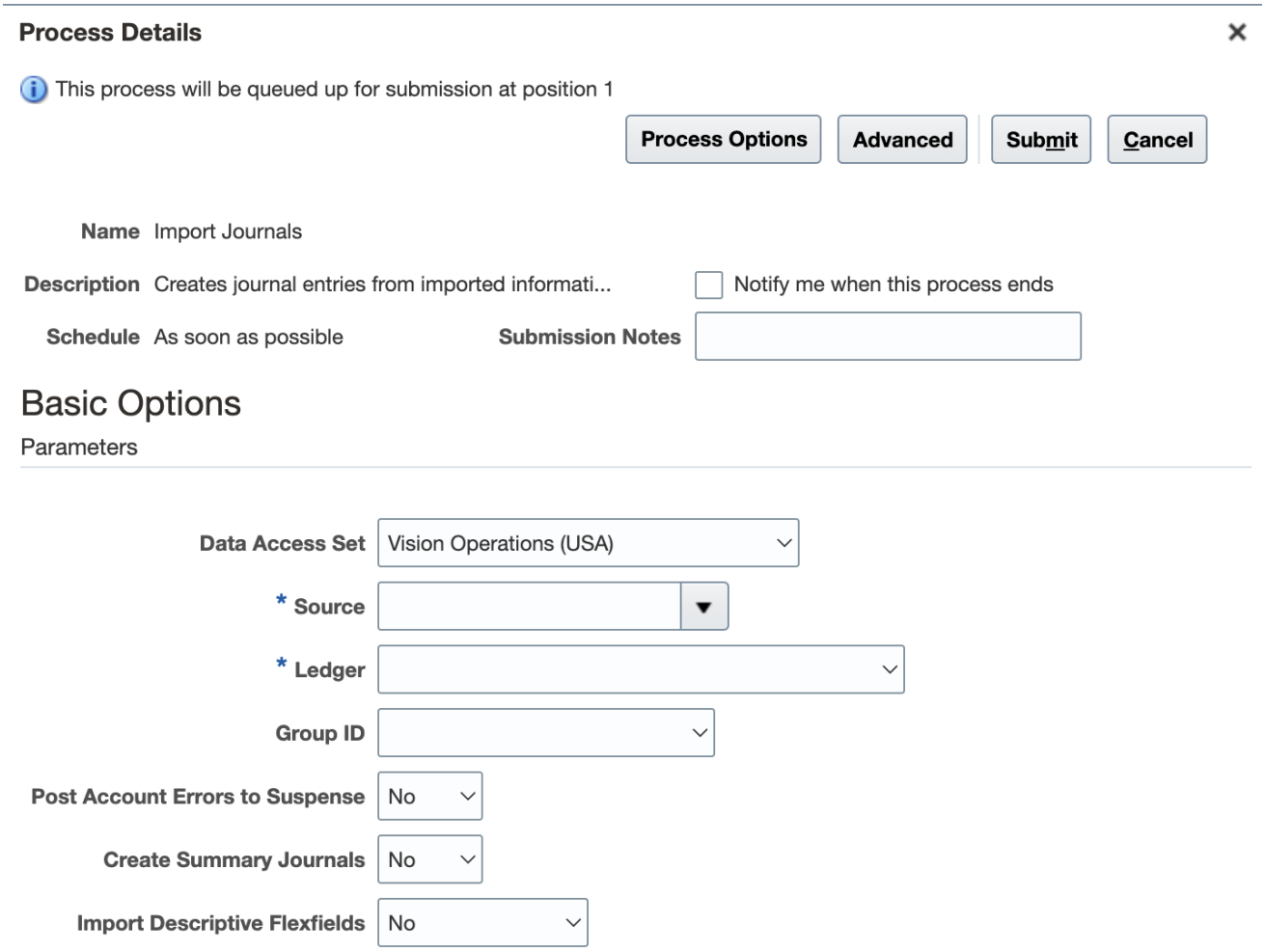
5. Select and Click "**Manage Enterprise Scheduler Job Definitions and Job Sets for Financial, Supply Chain Management, and Related Applications**" and **search** for Job name like "%Import Journal%".



6. Select the "**Import Journals**" row and click the **Edit** icon **(**please do not change anything as our goal is to only get job package and name**).**



7. From the above image, the job package is **PATH** and job name is **NAME**. These will be the first two columns in your property file for "Import Journals" with third column the sample prefix name of the zip file as suggested earlier:
oracle/apps/ess/financials/generalLedger/programs/common/,JournalImportLauncher,GL_BU_ABC,
<list of parameter separated by comma>
8. In the above image, the parameters details can be seen at the bottom half. Alternatively, parameters details can be obtained by following these steps:
    a. Open the Fusion Apps main page.
    b. Click the **Navigator**, and then click **Scheduled Processes in Tools**.
    c. Click on "**Schedule New Process**" button and search (and select) for "**Import Journals**". It will display the list of parameters as shown below:



Appendix 19: ExportBulkData Operation Precise ExtractFileType Support

Introduction

ERP Integration Service can extract data from Oracle ERP Cloud through Business Intelligence Publisher (BIP) tool using the ExportBulkData operation.

ExportBulkData supports extraction of .CSV, .XML & .TEXT type OUT files and .LOG files as a single ZIP named as ExportBulkData_<JobName>_<RequestIdReceivedInExportBulkDataResponse>.zip.

Until Update 22A, users had no choice over the type of files that ExportBulkData extracts i.e. the ZIP file contained all supported files (.CSV, .XML, .TEXT, & .LOG), if present. This inability to extract precise output file was leading unnecessary extraction of unwanted OUT/LOG files, consuming additional runtime and download time. For example, even if user needs only CSV OUT file, TEXT and XML OUT files, and LOG file are extracted (If present).

Starting update 22B, users will be able to specify a precise ExtractFileType from the list of supported file types and extract only those files.

This will provide more flexibility to the user and reduce significant runtime of integration by skipping download of files that are not needed.

Steps to Enable ExtractFileType Support

Configure Opt In

To enable the ExtractFileType support, configure the following Lookups using the **Manage Standard Lookup** page.

- Lookup Type: **FUN_OPTIN_WS_CONTROL**
- Lookup Code: **ADVANCE_FILE_TYPES**

Pass ExtractFileType in Request Payload

Once the Dev Opt In **FUN_OPTIN_WS_CONTROL/ADVANCE_FILE_TYPES** is configured, user can pass the required ExtractFileType in the **JobOption tag** in the request payload as **ExtractFileType=<RequiredFileTypes>**

- Providing the **ExtractFileType** in **JobOption tag** is not mandatory. If **ExtractFileType** is not provided, both OUT(.CSV, .XML, .TEXT) and LOG files will be extracted.
- If the Dev Opt In FUN_OPTIN_WS_CONTROL/ADVANCE_FILE_TYPES is not configured, both OUT(.CSV, .XML, .TEXT) and LOG files will be extracted even if **ExtractFileType** is passed in **JobOption tag.** The provided **ExtractFileType** is ignored.
- A combination of more than one valid **ExtractFileType** is also supported. Pass **ExtractFileType** as semi-colon separated valid file type value provided in table below:

Providing an optimum and precise *ExtractFileType* combination is recommended for best performance.

ExportBulkData Operation Payload with Precise ExtractFileType

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:typ="http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationService/types
/">
  <soapenv:Header/>
  <soapenv:Body>
    <typ:exportBulkData>
      <typ:jobName>JOB_PATH,JOB_NAME</typ:jobName>
      <typ:jobOptions>ExtractFileType=PASS_SUPPORTED_FILE_TYPES_HERE</typ:jobOptions>
      <typ:callbackURL>#NULL</typ:callbackURL>
      <typ:notificationCode>#NULL</typ:notificationCode>
    </typ:exportBulkData>
  </soapenv:Body>
</soapenv:Envelope>
```

ExportBulkData Operation Payload with Precise ExtractFileType: Example

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:typ="http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationService/types
/">
  <soapenv:Header/>
  <soapenv:Body>
    <typ:exportBulkData>
      <typ:jobName>/oracle/apps/ess/custom/shared/Custom/Financials/,TaxReportEssJob</typ:jobName>
      <typ:jobOptions>ExtractFileType=CSV;ZIP_ONLY</typ:jobOptions>
      <typ:callbackURL>#NULL</typ:callbackURL>
      <typ:notificationCode>#NULL</typ:notificationCode>
    </typ:exportBulkData>
  </soapenv:Body>
</soapenv:Envelope>
```

Oracle Recommendation

Oracle recommends using the CSV format for the downstream processing and extracted output must be a compressed ZIP file.

It will ensure optimal performance and seamless integration.

Recommended ExtractFileType = **CSV;ZIP_ONLY**

Supported ExtractFileType Combinations

Either pass a valid combination of supported ExtractFileType in the Job Option or do not pass the ExtractFileType at all.

exportBulkData operation returns an exception for any invalid ExtractFileType combination.

Sl. No.	Valid ExtractFileType Combinations
1	CSV
2	XML
3	TEXT
4	LOG
5	OUT
6	ALL
7	CSV;XML
8	CSV;TEXT
9	CSV;LOG
10	CSV;ZIP_ONLY **(Recommended)**
11	XML;TEXT
12	XML;LOG
13	XML;ZIP_ONLY
14	TEXT;LOG
15	TEXT;ZIP_ONLY
16	OUT;ZIP_ONLY

Sl. No.	Valid ExtractFileType Combinations
17	ALL;ZIP_ONLY
18	CSV;XML;TEXT
19	CSV;XML;LOG
20	CSV;XML;ZIP_ONLY
21	CSV;TEXT;LOG
22	CSV;TEXT;ZIP_ONLY
23	CSV;LOG;ZIP_ONLY
24	XML;TEXT;LOG
25	XML;TEXT;ZIP_ONLY
26	XML;LOG;ZIP_ONLY
27	TEXT;LOG;ZIP_ONLY
28	CSV;XML;TEXT;LOG
29	CSV;XML;TEXT;ZIP_ONLY
30	CSV;XML;LOG;ZIP_ONLY
31	CSV;TEXT;LOG;ZIP_ONLY
32	XML;TEXT;LOG;ZIP_ONLY
33	CSV;XML;TEXT;LOG;ZIP_ONLY

Notes:

- ExtractFileTypes CSV, XML, TEXT and LOG refers to .CSV, .XML, .TEXT and .LOG files respectively .
- ExtractFileTypes ALL refers to a combination of all OUT files and LOG file (.CSV + .XML + .TEXT + .LOG).
- ExtractFileTypes OUT refers to a combination of all OUT files (.CSV + .XML + .TEXT).
- ExtractFileTypes ZIP_ONLY optimises the performance by skipping the raw file upload to UCM.

ORACLE

Appendix 20: ImportBulkData Operation with Zip Operation

Introduction

For various inbound integrations, multiple data files required to run a 'Load Interface File for Import' job are already uploaded to UCM as individual uncompressed files from various sources (reporting Integrations, output of other jobs/Integration etc.). *For example*, a *purchase order integration will generate import payables invoice headers and import payables invoice lines data into multiple CSV files and upload to UCM.*

Until Update 23A, users had to download these files and compress them at the client site, then run the "Load Interface File for Import" job to load the data in Interface table(s). Hence, the same limitation also applies to importBulkData, it supports only a single documentID.

Starting Update 23B, 'importBulkData' introduces a new feature that allow users to provide multiple documentIDs for uploading data into Interface table(s).

Details:

The new feature can be enabled by passing '**ZipOptions=ZipAndSubmit'** in **<JobOption>** tag along with comma separated DocumentIds, InterfaceDetails:

- ImportBulkData downloads the files corresponding to the provided documentIDs from UCM , compresses them into a ZIP file and uploads to the UCM DocumentAccount corresponding to the provided InterfaceDetails(InterfaceOptionId).
- Post this, ImportBulkData retains the existing behaviour:
    - Submits the 'Load Interface File for Import' job and returns the "RequestID".
    - ImportBulkData also submits the Import Job(s), 'Upload Interface Error and Job Output File to Universal Content Management' job, and 'Purge Interface Tables' job as specified in the request payload.

Critical Considerations:

1. DocumentIds, Document Title and InterfaceDetails (In Job Option) are **mandatory** when a valid ZipOptions as 'ZipAndSubmit' is provided.
2. Provided documentIDs must not be duplicated for a single request.
3. Document title of the files corresponding to the provided documentIDs must not be duplicated.
4. Provided documentIDs must be of a 'Load Interface File for Import supported file of format document' such as '.csv', '.txt', '.xml', '.dat' and '.ack'.
5. Number of files in ZIP should match number of documentIds provided.

Request Payload: with ZipOptions=ZipAndSubmit

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns1:importBulkData
xmlns:ns1="http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationService/types
/">
      <ns1:document
xmlns:ns2="http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationService/">
        <ns2:DocumentTitle>APTEST_09012022</ns2:DocumentTitle>
        <ns2:DocumentAuthor>TestUser1</ns2:DocumentAuthor>
        <ns2:DocumentId>82077,82079</ns2:DocumentId>
      </ns1:document>
      <ns1:jobOptions>ZipOptions=ZipAndSubmit,InterfaceDetails=1,ExtractFileType=ALL</ns1:jobOptions>
```

```
    </ns1:importBulkData>
  </soap:Body>
```

Figure 60: Sample request payload for importBulkData with ZipOptions as ZipAndSubmit

Response Payload: With ZipOptions=ZipAndSubmit

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsa="http://www.w3.org/2005/08/addressing">
  <env:Header>

<wsa:Action>http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationService/ErpI
ntegrationService/importBulkDataResponse</wsa:Action>
    <wsa:MessageID>urn:uuid:b9159e2a-dc9b-4f1f-aaf0-55669e57eb63</wsa:MessageID>
  </env:Header>
  <env:Body>
    <ns0:importBulkDataResponse
xmlns:ns0="http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationService/types
/">
                    <!--RequestID of the Load Interface File for Import job-->
      <result
xmlns="http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationService/types/">3
6954</result>
    </ns0:importBulkDataResponse>
  </env:Body>
</env:Envelope>
```

Figure 61 : Sample response payload for importBulkData with ZipOptions as ZipAndSubmit

Appendix 21: Supported Import Processes by Load Interface File for Import Job (FBDI)

To get the details of supported File Based Data Import processes by the **Load Interface File for Import** job, use the **getInboundProcesses** operation of **ERPIcsIntegrationService(SOAP)**.

Refer to the following link for details: https://docs.oracle.com/pls/topic/lookup?ctx=fa-latest&id=OESWF
URL: https://<host:port>/fscmService/ErpIcsIntegrationService?WSDL

Example SOAP Request Payload

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:typ="http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationService/types
/">
  <soapenv:Header/>
  <soapenv:Body>
    <typ:getInboundProcesses/>
  </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Request Payload

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsa="http://www.w3.org/2005/08/addressing">
  <env:Header>
    <wsa:To>http://www.w3.org/2005/08/addressing/anonymous</wsa:To>

<wsa:Action>http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationService//ErpIcsIntegrationService/getInboundProcessesResponse</wsa:Action>
    <wsa:MessageID>urn:uuid:ece2e950-8516-4be1-98bc-049879852284</wsa:MessageID>
    <wsa:RelatesTo>urn:uuid:caabe942-ccd2-4790-96f8-2d1317b05f9b</wsa:RelatesTo>
  </env:Header>
  <env:Body>
    <ns0:getInboundProcessesResponse
xmlns:ns0="http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationService/types/">
      <result
xmlns="http://xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationService/types/">[
"InterfaceId":"15","JobDisplayName":"Import
Journals","UcmAccount":"fin/generalLedger/import","ImportJobName":"/oracle/apps/ess/financials/generalLedger/programs/common;JournalImportLauncher","JobDescription":"Creates journal entries from imported information from subledgers and other source systems."},{"InterfaceId":"1","JobDisplayName":"Import Payables Invoices","UcmAccount":"fin/payables/import","ImportJobName":"/oracle/apps/ess/financials/payables/invoices/transactions;APXIIMPT","JobDescription":"Creates Oracle Fusion Payables invoices from invoice data in the open interface tables."}....Similar info for all registered import processes]</result>
    </ns0:getInboundProcessesResponse>
  </env:Body>
</env:Envelope>
```

Alternatively, use the inboundProcesses operation of **ERPProcesses Service(REST) .** The ERPProcesses service can also provide the job parameter details.

Refer to this link for details: https://docs.oracle.com/en/cloud/saas/financials/23c/farfa/api-erp-processes.html

Appendix 22: Export Financials Data without Using Callback

This appendix focuses on alternative approach to design an integration for exporting financials data without callback.

1. Use the export bulk data operation to implement high volume data extraction and reporting solution. Call the ExportBulkData operation of ERP integrations REST resource to initiate the export.
    - ExportBulkData operation runs the reporting job and automatically submits an instance of "Upload Interface Error and Output Details to UCM" job that extracts, compresses, and uploads the output and logs to UCM.
    - The uploaded ZIP file name is always ExportBulkData_<JobName>_<RequestIdReceivedInExportBulkDataResponse>.zip which is uploaded to the Attachment Security group.
2. Design an integration to call the getDocumentsForFilePrefix operation in loop until a document ExportBulkData_<JobName>_<RequestIdReceivedInExportBulkDataResponse>.zip is received.
3. In the REST version of ERP Integration Service getDocumentsForFilePrefix operation is not available. Instead use the following.
    - Design an integration to call the getDocumentIdsForFilePrefix operation in loop until a documentID is received.
    - Use getDocumentForDocumentId operation to download the ZIP file by passing the documentID received.

Acronym

Abbreviation	Full Form/ Meaning
Abbreviation	**Full Form/ Meaning**
UCM	Oracle Universal Content Management
ADFdi	Application Development Framework Desktop Integration is part of Oracle's Application Development Framework that integrates with MS Excel to load moderate amounts of data into Oracle Cloud
ATK	Oracle Fusion Application Toolkit
ESS	Enterprise Scheduling Service

References

BI Publisher	https://<host:port>/xmlpserver For specific report: http://<host:port>/xmlpserver/<ReportDirectory>/<ReportName>.xdo Path to access from ATK : Navigator > Tools > Reports and Analytics > Create
ATK Home page	https://<host:port>/fscmUI/faces/FuseOverview