Oracle® Fusion Middleware

Oracle WebCenter Forms Recognition AP Project Guide

Version 1007G

E50184-02

November 2015

Documentation for the WebCenter Forms Recognition AP Packaged Project, that describes the features and configuration of the project.

ORACLE®

Oracle Fusion Middleware Oracle WebCenter Forms Recognition AP Project Guide, 11*g* Release 1 (11.1.1)

E50184-02

Contents

1 Introduction

1.1 Purpose

This document, intended for use by a trained WebCenter Forms Recognition implementer, is designed as a product user guide for the AP Packaged Project, version 1007G. It provides an overview of the solution, and explains the project configuration required for a production environment without the implementer needing to be a WebCenter Forms Recognition developer or Forms Recognition scripting developer.

The document is semi-technical in nature and requires the implementer to have an understanding of business processes.

1.2 Scope

This document provides the user with an understanding of the project configuration (.ini file) settings. For the successful implementation of the solution, it is recommended that the implementer be familiar with the following documentation, which is complementary to this document and the AP Packaged Project:

- *WebCenter Forms Recognition Installation Guide*
- *WebCenter Forms Recognition AP Project Migration Guide*
- *WebCenter Forms Recognition AP Project Release Notes*
- *WebCenter Forms Recognition Runtime Server User's Guide*

Note: The above reference documentation must be equivalent to the WebCenter Forms Recognition version and the AP Packaged Project version of the solution being deployed.

2 Solution Overview and Scope

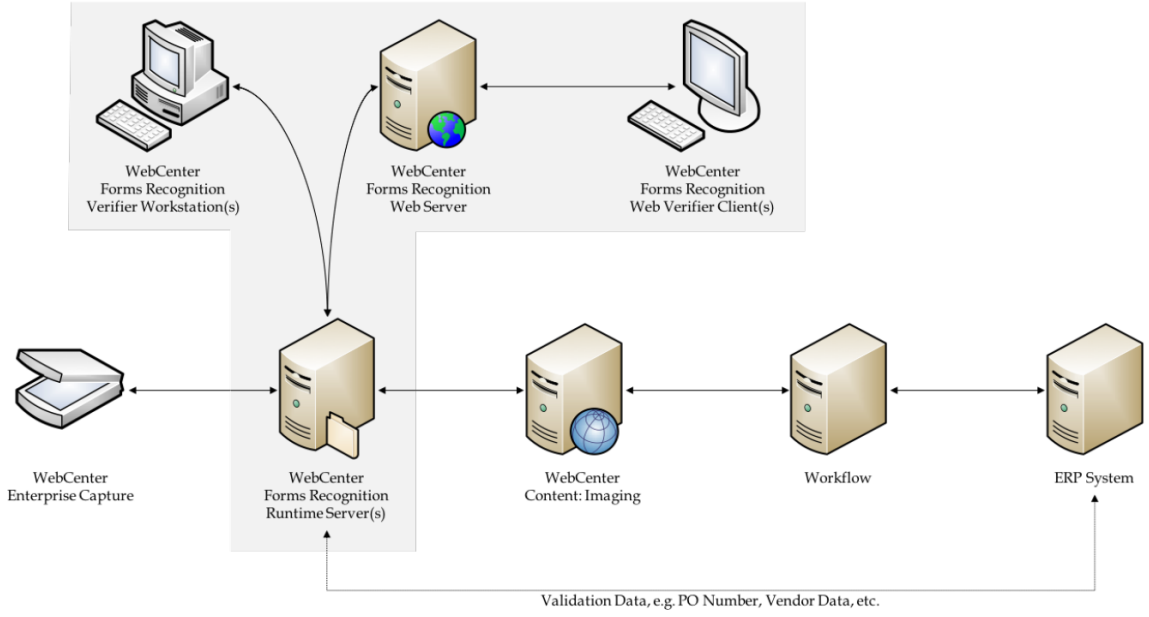
The AP Packaged Project is a pre-developed WebCenter Forms Recognition project to automate data entry for PO-based and non-PO based invoices, credit notes and third party freight invoices.

2.1 Typical Invoice Processing Solution Architecture

The diagram below illustrates a typical architecture for an invoice processing solution.

Invoice documents are scanned or otherwise captured and passed to the Runtime Server. The invoice document batches are processed by WebCenter Forms Recognition using the AP Packaged Project, with the document images and metadata being subsequently archived to an ECM system, for example Oracle WebCenter Content: Imaging. The Forms Recognition Verifier workstations are used for document quality assurance. Once archived to the ECM system, the documents are processed through a workflow, which is used for escalating of exceptions and other AP functions prior to the invoice being imported into the ERP system. Oracle Forms Recognition performs data validation from the ERP system.

Note: This architecture diagram and the accompanying description are provided for example reference purposes only. It does not represent any specific product or architectural requirements of WebCenter Forms Recognition or the AP Packaged Project.



2.2 Prerequisites

Version 1007G of the AP Packaged Project requires Oracle WebCenter Forms Recognition 11.1.1.8.0 or later, although version 11.1.1.9.0 using the FineReader 11 OCR engine is recommended.

2.3 Supported AP Documents

The AP Packaged Project currently supports the following document types:

- Vendor invoices
- Vendor credit memos
- Third party freight invoices

Additional document types, such as statements or travel & expense forms, will need to be configured within the solution as new document classes, if required.

2.4 Supported Languages

The AP Packaged Project is a language-independent solution and, out of the box, is able to process documents using the Western character set, the Cyrillic character set, Simplified Chinese, Japanese, Korean and Greek.

The AP Packaged Project has been specially optimized to increase extraction rates for invoices presented in English, Dutch, German, French, Spanish, Portuguese, Polish, Czech, Lithuanian, Latvian, Estonian, Turkish, Danish, Finnish, Norwegian, Swedish, Slovenian, Greek, Romanian, Bulgarian, Russian, Hungarian, Simplified Chinese, Japanese, Korean and Italian.

2.4.1 System Locale Considerations

The AP Packaged Project runs independent of whether the Windows locale settings use a period character or a comma as the decimal separator, even if the server is set to one option, yet one or more Verifier stations use something different.

For locales that use a space as the thousand separator (for example, the French locale) this must be changed to a comma or period character, whichever is appropriate.

Dates are handled by the project internally, in a manner that is entirely independent of the system locale. The Verifier display and output formats are configurable within the project.

Amount fields are outputted using a period characters as the decimal separator in all instances. If database output is required, the language and decimal separator preferences against that database should be configured accordingly.

2.5 Project Configuration

Project requirements are configured via the project configuration (.ini) file. These settings are dominant to the property settings with the AP Packaged Project itself, but recessive to those configured within the WebCenter Forms Recognition Runtime Server for the defined project. The solution permits the configuration of the following:

- Business rules relating to pre-defined data fields and document scope.
- Database connection settings for validations and reporting via the AP Project Reporting tables.
- Connection settings to ERP systems and other external data sources.
- Data export settings.
- Document archiving and export image file format.
- Error messages to be displayed within WebCenter Forms Recognition Verifier.

- Tax code validation and determination.

Refer to *Section 4: Configuration Settings* for a comprehensive description of all of the available configuration settings.

2.6 Recommended Registry Settings

The following registry settings are recommended when using the AP Packaged Project:

- AnalyzeLinesOptionally
- ASEnginePoolAllowedCharDifference

AnalyzeLinesOptionally provides optimal performance when the system is calculating the orientation of each text line from the OCR results.

ASEnginePoolAllowedCharDifference is recommended for projects where line pairing and vendor identification using the Associative Search Engine are being used. By default, WebCenter Forms Recognition will disregard near-identical vendor entries and PO lines with near-identical descriptions, which is often undesirable. Adding this registry key will ensure that the system returns all relevant records.

Both of the registry values described above should be created as **DWORD (32-bit)** values under the HKEY_LOCAL_MACHINE\SOFTWARE \[Wow6432Node \]Oracle\Cedar key as follows:

```
AnalyzeLinesOptionally                  REG_DWORD    0x00000001 (1)
ASEnginePoolAllowedCharDifference       REG_DWORD    0x00000000 (0)
```

Note: These settings are added to the Windows registry automatically during the installation of WebCenter Forms Recognition.

2.7 Web Verifier Client Settings

If the Web Verifier client is to be deployed, the **web.config** file located in the <*Installation Folder*>**\WebCenter Forms Recognition Web Server** directory needs to be updated so that the dialog box information and option popups are activated within the Web Verifier interface.

In the **web.config** file, the *mouseClicked enabled* and *tabPressed enabled* properties should both be set to **true** as shown below:

```
<formEvents>
    <mouseClicked enabled="true"/>
    <tabPressed enabled="true"/>
    <itemCopied enabled="false"/>
    <tableCellSelected enabled="false"/>
</formEvents>
```

2.8 Activating Asian Language Recognition

If the system is required to process documents presented using the Chinese, Japanese or Korean language sets, the following additional configuration must be performed.

2.8.1 Activate CJK Recognition Within the Project

1. Open the project in the Designer application
2. Select the **Settings…** option from the **Options** menu. The **Settings** dialog will open

3. Select the **Definition Mode** tab
4. Check the **Activate support of non-western languages** checkbox
5. Check the **Use multi-byte encoding** checkbox
6. Click **OK** to close the **Settings** dialog
7. Save the project

2.8.2 Add the Required OCR Language

8. Select **Definition Mode** from the **View** menu
9. Select the project-level node (**AP Packaged Project_1007G.sdp**) on the **Classes** tab
10. Select the **Show Properties** option from the **Edit** menu
11. Click the **OCR settings…** button on the **Project** tab of the properties pane. The **OCR Properties** dialog will be displayed
12. Select the **Recognition** tab
13. Select the **Languages** sub-tab. The list of installed FineReader languages will be displayed beside the list of languages currently used by the project. (The default is **Digits** and **English**)
14. Use the arrow buttons to add or remove the desired languages between the **Installed** and **Used** lists

---

Note: Ensure **Digits** is present in the **Used** list.

Ensure that the <*language*>**+English** language is used. For example, if Japanese is to be used, select **Japanese+English**.

---

15. Click **OK** to accept the configured OCR settings and close the dialog
16. Save the project

2.8.3 Add the Required Registry Setting

17. Using the Registry Editor, add a new key called **CJKT Support** under the existing **HKEY_LOCAL_MACHINE\SOFTWARE***[Wow6432Node\]***Oracle** key
18. Create a new **DWORD (32-bit)** value within the **CJKT Support** key, called **CJKT_MinimalSymbolSequenceLengthForWordsSplit**
19. Set the value data to **1** as in the example given below:

```
HKEY_LOCAL_MACHINE\SOFTWARE \Wow6432Node\Oracle\CJKT Support
  ▪ CJKT_MinimalSymbolSequenceLengthForWordsSplit   REG_DWORD      0x00000001 (1)
```

Note: For optimum OCR performance, it is recommended that projects involving documents with CJK characters be processed via a separate instance of the project file.

---

3 Project Fields and Features

The following section provides detail on the standard fields delivered in the AP Packaged Project version 1007G, and the additional features and integration options that are available.

3.1 Extraction Fields

3.1.1 Document Type

The Document Type field denotes whether the incoming document is either an invoice or a credit memo. The field result is determined by the system automatically, but may be changed within the Verifier application via the drop-down.

Using the project configuration options, the system administrator may enter indicative words and phrases for a credit memo that will influence the document type selection. The system default value is **INVOICE**.

3.1.2 Invoice Type

The Invoice Type field denotes whether the invoice is purchase order (**PO**) or non-purchase order related (**NO-PO**).

The invoice type determines:

- Whether line items are required from the invoice
- How the invoice should be handled downstream
- Whether a purchase order number is required

Within the project configuration settings, a default value for this field can be configured. In an environment where the majority of invoices are purchase order related, it makes sense to set this default to **PO**.

Further configuration options are available to:

- Overwrite a **NO-PO** default and set to **PO**, based upon whether a valid purchase order number is detected.
- Overwrite the default based upon an attribute in the ERP vendor master data that would indicate whether non-PO invoices from this vendor are permitted.
- Overwrite the default based on a component of the image filename set by the scanning software, where PO and non-PO invoices are pre-sorted and scanned using different scan jobs.

3.1.3 PO Type

The PO Type field denotes whether the extracted purchase order relates to materials or services.

The default is **MATERIAL**, but it is possible to configure the project to switch this to **SERVICE** depending on the following purchase order characteristics:

- The purchase order document type.
- The line type or item category of the purchase order lines.

- The unit of measure on the purchase order lines.
- The prefix of the extracted purchase order number.

The content of the PO Type field controls whether line items are required, whether just the total of each line item and a description is required, and how the system handles the invoice during the line pairing event at document export.

Note: Line pairing does not support purchase orders that contain a mixture of material and service line items.

The PO Type is set from the purchase order in accordance with the configuration settings in the *PON section* of the project configuration.

3.1.4 Invoice Number

The Invoice Number field is mandatory for all documents, but the project can be configured to accept a blank or invalid invoice number in the *NUM section* of the project configuration, if the invoice is from a utility vendor.

If the invoice is not from a utility vendor where an invoice number is not required, the system will set it to invalid if:

1. OCR errors are detected in one or more of the characters where the confidence level falls below the required minimum (default 50%).
2. The system has found more than one candidate on the document whose respective confidences are closer than the distance setting against the field (default 10%).
3. The format of the invoice number in terms of its length and sequence of alpha and numeric characters does not match previous invoice numbers submitted from the same vendor as stored in the invoice number validation table. This validation feature can be enabled and configured in the *NUM section* of the project configuration.

   Note: If enabled, this check is not carried out in the Verifier application as user input is assumed to be correct.

4. The *Duplicate Invoice Number Checking* feature is enabled, and a duplicate invoice number is found for the same vendor in the ERP system. This feature can be enabled and configured in the *WFR section* of the project configuration.

Configuration options also dictate how the invoice number will be formatted, with options available to:

- Remove all special characters from an extracted invoice number.
- Remove special characters if they appear at the start or end of the invoice number.
- Retain only a specified set of special characters.
- Remove spaces from within an invoice number.
- Remove any leading zeroes from the invoice number.

3.1.5 Invoice Date

The Invoice Date field is mandatory for all documents.

The system will automatically convert the invoice date on the document, irrespective of how it is expressed, into the designated Verifier output format, which can be set either to DD/MM/YYYY or MM/DD/YYYY via the configuration settings in *DAT section* of the project configuration.

This formatting relies on the vendor's country of origin being mapped and populated in the vendor master extract file in order to handle ambiguous dates. For example, **01/02/2009** is 2nd January 2009 in the US, but reads as 1st February 2009 in Europe.

If the vendor country is mapped in the *SRC section*, and this country exists in the list of countries where the national date preference is MDY (e.g. the US), then the system will convert to **01/02/2009** if the Verifier output format is **MM/DD/YYYY**, and to **02/01/2009** if the Verifier output format is **DD/MM/YYY**.

If a date is entered manually in the Verifier application, then no conversion will take place unless the date entered is impossible for the Verifier output format. For example, if the format is set to **MM/DD/YYYY** and the user enters **28/02/2009**, the system will automatically flip the date to **02/28/2009**.

The system can be configured to invalidate the invoice date if:

1. It is more than *x* days in the future.
2. It is not in the current month.
3. It falls more than *x* days prior to the current date, where *x* is configurable.

Machine and user local settings play no part in the system's internal handling of dates.

User input into the date field is not subject to the checks above as long as the date entered is valid for the output format.

If the downstream export event involves writing the extracted date into a flat file, or into a database table, the output format of the date can be set to **DDMMYYYY**, **MMDDYYYY** or **YYYYMMDD** with an optional separator.

The system is able to handle dates expressed in the Gregorian calendar and the Japanese Emperor's calendar. The Thai Buddhist calendar is also supported.

3.1.6 Company Code

The Company Code field represents the unique ID of the legal entity within the client's wider organization for which the invoice is intended. For implementations involving Oracle e-Business Suite, this field represents the organization ID; for implementations involving PeopleSoft, this field represents the accounts payable business unit.

System configuration settings determine whether this field is mandatory via the *CCO section* of the project configuration.

The field can be determined either:

1. Via a mapping from a component in the document filename set in the *IMP section* of the project configuration, if it is set at the point of scan.
2. Using the Associative Search engine pointing to a CSV or database extract of the master company code data, as specified in the *ASA section* of the project configuration.
3. Via a lookup to a database table or downstream ERP system based on the extracted purchase order number (PO invoices only). If a value is found and the system is configured to take the company code from the purchase order in all cases, this will over-write any company code determined by options 1 or 2 above.

The validity of user entry in this field can be checked against a database or a downstream ERP system as required. This is also set in the *CCO section* of the project configuration.

3.1.7 Vendor ID, Site ID and Internal Vendor ID

The AP Packaged Project employs its unique associative search engine in order to ascertain the invoice vendor.

By pointing the project to an extract of the client's vendor master, whether it resides in a flat file or in a database table, the system analyzes the text of the invoice, and then selects the closest matching vendor record in a fault-tolerant manner that accounts for spelling differences, OCR errors, abbreviations and vendor details embedded within logos on the invoice.

If the system is not confident enough that the closest matching vendor from the extract is the correct vendor, the field will be marked invalid, and the document sent to a Verifier. The Verifier user can choose either to accept this vendor, or they can select an alternative using the vendor search facility within the Verifier application.

The vendor ID is a mandatory field for both PO-based and non-PO invoices.

For PO-based invoices, the vendor is defaulted initially from the purchase order, although this setting can be disabled in the *VND section* of the project configuration, so that the vendor that the project has determined from the invoice takes precedence.

For ERP systems such as Oracle Financials, that use a vendor ID and a site ID, only the vendor ID component is used within the validation, and the vendor pay-to site does not have to be the same as the order-from site on the purchase order. The automatic extraction of the vendor will look for a vendor at a specific site. The site ID cannot be entered manually in the Verifier application, but is populated via the chosen result of the vendor search.

If the Verifier user wishes to select a vendor that is not represented on the purchase order (e.g. an alternate payee, a third party freight vendor), then this is possible as long as the vendor exists in the vendor master extract and an appropriate invalid reason is set.

For non-PO invoices, if the invoice vendor does not exist within the vendor master extract, then the invoice may only pass if an appropriate invalid reason is set.

The AP Packaged Project also supports scenarios where the ERP uses an external vendor ID for display to the user, but another vendor ID internally. In this scenario, the external vendor ID will be displayed within the Verifier application, but the system will store the internal vendor ID in the internal vendor ID field so that both values are available for export downstream.

3.1.8 Purchase Order Number and PO Extension

If activated, the purchase order number field is mandatory for all invoices where the invoice type is **PO**, unless an appropriate invalid reason has been selected.

The system will only extract a purchase order number if it matches a valid format as specified in the *PON section* of the project configuration. Further configuration options allow this field to be validated against a database table or against a downstream ERP system. If such a validation is set, the purchase order must exist in that system.

On the server side, an additional check is made to ensure that the pay-to vendor set against the extracted purchase order matches the vendor details on the invoice, but it is possible to configure the system to consider and validate the vendor ID and purchase order number independent of one another.

Within Verifier, the user may change the purchase order or vendor and the system will let them pass as long as the chosen vendor is referenced on the purchase order. If an alternate vendor is required, an appropriate invalid reason must be selected from the field drop-down. For ERP systems that use a site ID as well as a vendor ID to identify a unique vendor address, only the vendor ID component needs to be common between the vendor ID field and the purchase order details.

If the user changes the purchase order and the new purchase order does not contain the vendor currently set in the Vendor ID field, an information message will appear inviting the user to accept the new vendor or continue with the current vendor. A similar message will appear for the currency field if the currency is set to default from the purchase order.

The vendor ID and the purchase order can be entirely decoupled from one another via a setting in the *VND section* of the project configuration, so that the vendor that the project has determined from the invoice takes precedence.

If the new purchase order has not been released and the system does not require line items to be mandatory under this circumstance, a message will appear informing the Verifier user.

If the new purchase order is a one-to-one match with the invoice in terms of its overall value, or the value of goods received against the purchase order but not yet invoiced (MIRA scenario), and the system does not require line items to be mandatory under this circumstance, a message will appear informing the Verifier user.

If the purchase order number is missing or invalid, the Verifier user should select an appropriate invalid reason from the field drop-down box in order to progress the invoice through the system.

A purchase order must be present and valid for the line pairing feature to be activated during document export.

Multiple purchase orders on a single invoice are supported if activated in the *LPR section* of the project configuration, but the system will handle this at time of line pairing. From the point of view of extraction and operation in Verifier, only a single purchase order number is required.

The PO Extension field is populated in implementations involving JD Edwards or PeopleSoft. In JD Edwards implementations, this holds the purchase order type (e.g. **OP**); in PeopleSoft implementations, this holds the purchasing business unit.

3.1.9 Bill-to Name

The Bill-to Name represents the name of the legal entity for which the invoice is intended.

This field is mandatory in order to check that the incoming document is intended for a valid company within the client's organization. In Verifier, the user may pass a blank value by hitting enter in the field, thus confirming that the invoice is valid for the client to process.

Within the *BTO section* of the project configuration, the system administrator can specify words and phrases that anchor a valid bill-to name. At runtime, the full bill-to name will be extracted. If no anchors are specified or an appropriate anchor is missing, no value will be extracted into the field, hence all documents will stop in the Verifier application.

3.1.10 Invoice Subtotal

This field is used to capture the subtotal of the invoice.

This subtotal is generally not mandatory, but the system will convert any extracted value to a valid amount using a period/full-stop as the decimal separator, and will use this value in the validation calculation applied against all amount fields, namely:

Invoice Total = Line Item Total + Total Tax + Freight + Miscellaneous Charge – Discount – Withholding Tax – ISR Retention

If line items are not mandatory and have not been captured, then the amount fields can also be validated with the following:

Invoice Total = Subtotal + Total Tax + Freight + Miscellaneous Charge – Discount – Withholding Tax

The tolerance for the above calculations can be set in the *TOL section* of the project configuration.

The Subtotal field may be set as mandatory by setting the **SubtotalRequired** parameter to **YES** in the *AMT section* of the project configuration.

3.1.11 Invoice Freight Amount

This field is used to capture a freight charge specified by the vendor at header level. The system is also able to capture freight amounts at the line item level.

This field is generally not mandatory, but the system will convert any extracted value to a valid amount using a period/full-stop as the decimal separator, and will use this value in the validation calculation applied against all amount fields, namely:

Invoice Total = Line Item Total + Total Tax + Freight + Miscellaneous Charge – Discount – Withholding Tax – ISR Retention

If line items are not mandatory and have not been captured, then the amount fields can also be validated with the following:

Invoice Total = Subtotal + Total Tax + Freight + Miscellaneous Charge – Discount – Withholding Tax

The tolerance for the above calculations can be set in the *TOL section* of the project configuration.

If line pairing is activated to occur during document export, the system will gather up all invoice freight amounts, both in this field, and also at line item level, and will process them in accordance with the settings for such charges as specified in the *MSC section* of the project configuration. Standard options for processing include booking the freight as a planned or unplanned delivery cost, or creating a special general ledger entry against the invoice.

3.1.12 Invoice Miscellaneous Charge

This field is used to capture a non-freight miscellaneous charge specified by the vendor at header level (e.g. a fuel surcharge, administration charge, customs charge, pallet charge, etc.) The project is also able to capture miscellaneous charges at the line item level.

This field is generally not mandatory, but the system will convert any extracted value to a valid amount using a period/full-stop as the decimal separator, and will use this value in the validation calculation applied against all amount fields, namely:

Invoice Total = Line Item Total + Total Tax + Freight + Miscellaneous Charge – Discount – Withholding Tax – ISR Retention

If line items are not mandatory, then the amount fields can also be validated with the following:

Invoice Total = Subtotal + Total Tax + Freight + Miscellaneous Charge – Discount – Withholding Tax

The tolerance for the above calculations can be set in the *TOL section* of the project configuration.

If line pairing is activated to occur during document export, the system will gather up all miscellaneous charges, both in this field, and also at line item level, and will process them in accordance with the settings for such charges as specified in the *MSC section* of the project configuration. Standard options for processing include booking the miscellaneous charge as a planned or unplanned cost, or creating a special general ledger entry against the invoice.

If the customer requires each type of miscellaneous charge to be handled in a different manner, then a decision will need to be made which specific charge the miscellaneous charge header field represents, and the corresponding mapping between this field and charge type needs to be made in the project configuration settings.

The Verifier user should enter any other miscellaneous charges that appear as line items in the table.

3.1.13 Invoice Tax Amount

This field is used to capture the total invoice tax amount, such as US sales & use tax and European VAT. It is also used to capture Canadian GST/HST tax amounts and Brazilian IPI tax amounts.

This field is generally not mandatory, but the system will convert any extracted value to a valid amount and will use this value in the validation calculation applied against all amount fields, namely:

Invoice Total = Line Item Total + Total Tax + Freight + Miscellaneous Charge – Discount – Withholding Tax – ISR Retention

If line items are not mandatory, then the amount fields can also be validated with the following:

Invoice Total = Subtotal + Total Tax + Freight + Miscellaneous Charge – Discount – Withholding Tax – ISR Retention

The total tax is set to the value of the tax amount field plus the amount captured in the PST field, if any. Brazilian ICMS tax is not included in this calculation. The tolerance for the above calculations can be set in the *TOL section* of the project configuration.

If line pairing is activated to occur during document export, the system will attempt to determine the correct manner in which the tax should be booked in the downstream ERP system. Please refer to *Section 3.2.2: Automatic Tax Determination and Validation* for more information.

If the invoice is from a Canadian vendor, and Provincial Sales Tax (PST/QST) is captured on the invoice in the PST field, the tax amount exported will be the sum of the contents of the Tax field and the PST field. If the invoice is a Brazilian Note Fiscal, and ICMS tax is read from the document in the ICMS tax field, then this ICMS tax amount is also added to the total invoice tax amount exported.

3.1.14 Invoice Withholding Tax Amount and ISR Retention (Mexico)

The Withholding Tax Amount fields capture the portion of the invoice total amount that should be withheld by the client for legal reasons and not paid back to the vendor.

The ISR Retention field is used specifically to capture the ISR retention component of withholding tax, which can appear on Mexican invoices. The IVA retention component is captured in the regular withholding tax field.

These fields are not mandatory, but the system will convert any extracted values to a valid amount and will use these values in the validation calculation applied against all amount fields, namely:

Invoice Total = Line Item Total + Total Tax + Freight + Miscellaneous Charge – Discount – Withholding Tax – ISR Retention

If line items are not mandatory, then the amount fields can also be validated with the following:

Invoice Total = Subtotal + Total Tax + Freight + Miscellaneous Charge – Discount – Withholding Tax – ISR Retention

The tolerance for the above calculations can be set in the *TOL section* of the project configuration. The withholding tax base amount is set to the full invoice amount (i.e. Invoice Total + Withholding Tax) minus the total invoice tax amount.

At time of export, withholding tax is outputted as a separate header level field with the ISR retention amount added to it. The ISR retention field is also available as a separate export parameter. The invoice total amount is outputted with the withholding tax and ISR retention amounts added back on.

3.1.15 Provincial Sales Tax (PST/QST)

This field is used to capture the Provincial Sales Tax (PST/QST) component of Canadian tax.

The system will attempt to extract the PST/QST amount from a document if the vendor country of origin is Canada. The regular invoice tax amount field is used to capture the GST component of Canadian tax.

During the mathematical validation of the invoice amounts, both the PST/QST and regular invoice tax amounts are added together in the background to form the total tax liability of the invoice. At time of document export, the tax amount exported is the sum of the regular tax field and the PST field, although the PST component is available separately.

For example, if, on an invoice, the GST component is 40 CAD, and PST/QST is 10 CAD, the total tax amount will be exported as 50 CAD, and the PST/QST amount will be exported as 10 CAD.

3.1.16 Invoice Header Discount Amount

This field is used to capture a discount given by the vendor at the invoice header level.

This field is not mandatory, but the system will convert any extracted value to a valid amount and will use this value in the validation calculation applied against all amount fields, namely:

Invoice Total = Line Item Total + Total Tax + Freight + Miscellaneous Charge – Discount – Withholding Tax – ISR Retention

If line items are not mandatory, then the amount fields can also be validated with the following:

Invoice Total = Subtotal + Total Tax + Freight + Miscellaneous Charge – Discount – Withholding Tax – ISR Retention

The tolerance for the above calculations can be set in the *TOL section* of the project configuration.

3.1.17 Invoice Total

This field is used to capture the total amount of the invoice.

This field is mandatory as long as an invalid reason designating otherwise has not been set, and its value cannot be zero.

The system will convert any extracted value to a valid amount and will use this value in the validation calculation applied against all amount fields, namely:

Invoice Total = Line Item Total + Total Tax + Freight + Miscellaneous Charge – Discount – Withholding Tax – ISR Retention

If line items are not mandatory, then the amount fields can also be validated with the following:

Invoice Total = Subtotal + Total Tax + Freight + Miscellaneous Charge – Discount – Withholding Tax – ISR Retention

The tolerance for the above calculations can be set in the *TOL section* of the project configuration.

It is possible to decouple the validation of the invoice total amount from the values of other fields by setting the **ValidateTotalOnly** parameter to **YES** in the *AMT section* of the project configuration.

3.1.18 Currency

The Currency field contains the ISO code of the invoice currency.

In the first instance, the AP Packaged Project will attempt to extract the currency from the invoice. In the event that no currency is captured or no currency appears on the invoice (which is common for domestic transactions) the currency field can be set to default either to the currency associated with the vendor's country of origin, or the currency in which the purchase order was raised.

Configuration settings determine whether the currency is mandatory or not, and also whether user input should be validated against a database or the downstream ERP system.

Note: The **WFR_AP_Tables_Create_<***database***>.sql** script provided with the AP Packaged Project creates and populates a table called **OFRCOUNTRY** that provides data for all world currencies and their associated countries.

Within the configuration, it is also possible to specify which currency symbols and terms are associated with each individual currency, for example, **pounds**, **sterling** and **£** are associated with **GBP**. At runtime, if the corresponding currency symbol is found and this symbol is unique to one particular currency, then this is the currency that will be selected. Terms take priority over currency symbols. Additionally, the higher the currency is in the list in the project configuration, the greater weight the system will attach to this currency.

If the currency symbol is ambiguous, then the vendor's country of origin will be used to find the corresponding currency ISO code. For example, the **$** currency symbol is found on the document and the vendor is from Canada, hence the project will set the currency to **CAD** for Canadian dollars.

3.1.19 Bank Account Number and Bank Account Code

The Bank Account Number field is used to capture the bank account into which the vendor has requested payment to be made. The Bank Account Code field represents the identification of that bank account for that vendor from the point of view of the downstream ERP system.

The bank account is determined based on whether the bank details attribute is mapped and populated from the vendor extract file. This is done within the *SRC section* of the project configuration. For each account specified where the account currency matches the currency of the invoice, the system will look for the bank details on the document. If they are found, the first matching bank account number and the corresponding bank account code will be copied into the fields.

If the user enters a new purchase order or vendor within the Verifier application, the bank account details will be re-assessed automatically by the system.

It is possible to limit the identification of bank accounts only to those vendors who require payment to be made via a bank transfer. If the payment methods field is mapped and populated for the vendor, and the list of payment methods denoting a bank transfer is in place in the *PMT section* of the project configuration, then the system will only look to extract a bank account if the list of vendor payment methods contains an entry that denotes a bank transfer as a possible payment method.

3.1.20 Payment Order Reference (POR) Number and POR Subscriber Number

The Payment Order Reference Number is a 27-character transaction ID applied to the invoice by the Swiss Postal Service.

The AP Packaged Project will extract this value from the document (typically domestic invoices supplied by Swiss vendors) and place it in the POR Number field.

The POR number will only be passed downstream during data export if the vendor has a POR subscriber number mapped and available within the vendor master data extract, or if a POR subscriber number has been extracted from the document. This subscriber number is mapped in the *SRC section* of the project configuration. The POR subscriber number in the vendor master takes priority over a POR subscriber number extracted from the invoice.

Neither the POR Number nor the POR Subscriber Number are mandatory.

3.1.21 Payment Reference

This field is used to capture the vendor's payment reference as specified on the invoice. It is not mandatory.

The Payment Reference is used in the Nordic countries of Finland, Sweden and Norway. In Norway, for example, it is referred to as the KID number.

By default, the extraction of this field is deactivated. Refer to *Appendix I: Activating Optional Project Fields*.

3.1.22 Exchange Rate and Local VAT Amount

If the VAT compliance check is activated in the *TAX section* of the project configuration, then an exchange rate or VAT amount in local currency must be entered if VAT is being charged in a currency that is not the local currency of country where VAT is being levied.

The exchange rate should be the value that the invoice tax amount should be multiplied by to get the same tax amount in the local currency.

At time of export, only the exchange rate is passed downstream. If a local VAT amount was entered, the system will calculate the exchange rate from the invoice currency to the local currency automatically.

3.1.23 Account Number

The Account Number field represents the unique identification number of the client from the point of view of the vendor.

This is a mandatory field in lieu of the invoice number for invoices from utility vendors if the project configuration is set to skip invoice number extraction for utility vendors. A vendor is marked as being a utility vendor via the mapping in the *SRC section* of the project configuration, and the value in the vendor master extract column contains the positive value for a utility vendor as specified in the *NUM section*. In all other cases, the field is not mandatory.

3.1.24 Priority Flag

The Priority Flag field is set to **YES** or **NO** depending on the urgency of processing.

The value of this field defaults to **NO**, but can be overwritten either by:

1. The Verifier user via the field drop-down, or;
2. A component in the document filename being mapped to the priority flag in the *IMP section* of the project configuration, and the value of that component matches the positive value for the priority flag.

At point of document export, this value can be passed to the downstream workflow so that the item can be prioritized accordingly.

To increase the item priority in the AP Packaged Project, documents should be sorted according to priority during the scanning process, and then outputted to a different AP Packaged Project import directory, which is processed by an RTS instance that sets the priority of all imported documents to **1**.

3.1.25 Scan Date

The Scan Date field represents the date on which the invoice was scanned.

This is not extracted from the document, but is set via a mapping to the field from the document filename. The expected format of the date lifted from the document filename can be configured within the *IMP section* of the project configuration. The system will subsequently convert the date into the Verifier output format.

If the downstream export event involves writing the scan date into a flat file, or into a database table, the output format of the date can be set to **DDMMYYYY**, **MMDDYYYY** or **YYYYMMDD** with an optional separator as configured in the *EXP section* of the project configuration.

3.1.26 Batch Name

This Batch Name field represents the name of the batch into which the invoice was scanned.

This is not extracted from the document, but is set via a mapping to the field from the document filename.

3.1.27 Unique Reference Number (URN)

The URN field is the unique reference number assigned to the document in the upfront scanning process.

This is tied to the project field via a mapping from the document filename. If the field is not mapped to a specific filename component, then the value of the URN field is set by the system to be the entire document filename minus the path and file extension.

The URN can be used by the AP Packaged Project in order to:

1. Set a key for the document record within the database reporting.
2. Set a key for the document record for the purposes of database export, and a unique filename for the purposes of flat file export.
3. Denote the unique archive document ID for the image as determined by an early archiving process.

3.1.28 Invalid Reason

The Invalid Reason field contains a list of possible exceptions that could prevent a Verifier user from being able to correct a document in its entirety.

The system default is **NONE**, but a Verifier user may change this value through the field drop-down when a particular exception is encountered, so that the document may be progressed out of the Verifier application.

The following table contains a list of the system delivered invalid reasons, their corresponding rule, when they should be selected, and the effect of selecting them:

Invalid Reason	Usage	Effect
VENDOR NOT FOUND	This invalid reason should be selected if the invoice vendor cannot be found using the vendor search function. This applies to both PO-based and non-PO invoices.	RULE: SETVENDORTOVALID The vendor ID field is set to valid. In the *TAB section* of the project configuration, it can be decided whether line items are still required or not. If the export event has been configured to create documents directly in a downstream ERP system, document export will fail for non-PO invoices, as the ERP system will not permit an invoice to be created without a vendor ID. Line pairing will not be carried out at the time of document export. If activated, the VAT registration compliance check will not be carried out. No vendor details will be exported.
MISSING/INVALID PO	This invalid reason should be selected if the invoice is purchase order related but the vendor has either failed to quote a purchase order number, or the purchase order number they did quote was invalid for the invoice.	RULE: SETPOTOVALID The purchase order number field is set to valid. In the *TAB section* of the project configuration, it can be decided whether line items are still required or not. Line pairing will not be carried out at the time of document export. If activated, the VAT registration compliance check will not be carried out.
MISSING/INVALID VENDOR AND PO	This invalid reason should be selected if both the vendor and the purchase order are invalid or do not exist.	Rule: SETVENDORANDPOTOVALID The vendor ID, the PO number and line items are all set to valid. Line pairing will not be carried out at the time of document export. If the export event has been configured to create documents in a downstream ERP system, document export will fail.

Invalid Reason	Usage	Effect
		No vendor details will be exported. If activated, the VAT registration compliance check will not be carried out.
PO VENDOR IS NOT INVOICE VENDOR	This invalid reason should be selected if the user wishes to pass a different vendor ID to what is set against the purchase order.	Rule: ALLOWNONPOVENDOR The purchase order and vendor ID fields will both be set to valid providing the vendor exists in the vendor master data extract and the purchase order number passes validation.
INVOICE AMOUNTS DO NOT ADD UP	This invalid reason should be selected if the invoice is not mathematically correct and the figures do not add up within the specified tolerance.	Rule: SETAMOUNTSTOVALID All amount fields and all the line items will be set to valid. Line pairing will not be carried out. If activated, the VAT registration compliance check will not be carried out.
THIRD PARTY FREIGHT	This invalid reason should be selected if the invoice is from a 3rd party freight vendor quoting the material purchase order from another vendor where they have not been set-up as the vendor responsible for freight.	Rule: THIRDPARTYFREIGHT The vendor ID field will be set to valid as long as the vendor exists, and line items will not be required in Verifier. During line pairing, the net amount of the invoice will be posted either to unplanned delivery costs, to condition records, or to a general ledger account depending on the rules for the miscellaneous charge category assigned to third party freight vendors as specified in the *MSC section* of the project configuration.
NON VAT COMPLIANT	This invalid reason should be selected if the vendor has not complied with EU regulations that state that, if value added tax is to be charged, then both sets of VAT registration numbers (that of the vendor, and that of the bill-to party) must be stated on the invoice. The country prefixes of the VAT registration numbers must also be identical.	Rule: NONVATCOMPLIANT The vendor VAT registration number, bill-to VAT registration number, local VAT amount and exchange rate fields will be set to valid. Document export will run as normal with the invalid reason and its associated code being passed to the downstream system.
STOCK INVOICE	This should be used for PO invoices where the vendor legitimately does not quote a purchase order number on the document. For example, invoices that use retrospective purchase orders	Rule: STOCKINVOICE The PO number field will be allowed to pass blank, but all other fields require completing as normal. Instead, the purchase order number is decided programmatically at time of document export via user exit **UserExitSetPOForLinePairing** if line pairing is required.
ZERO VALUE INVOICE	This invalid reason should be selected if the invoice has a legitimate zero amount for the total.	Rule: ZEROVALUEINVOICE This will permit a zero value invoice total to pass as long as the overall invoice is in balance.
VENDOR ADDRESS INVALID	This invalid reason should be selected if the vendor can be found in the vendor search, but the vendor address does not	Rule: SETVENDORTOVALID The vendor ID field is set to valid. In the *TAB section* of the project configuration, it can be

Invalid Reason	Usage	Effect
	match what it on the invoice.	decided whether line items are still required or not. If the export event has been configured to create documents directly in a downstream ERP system, document export will fail for NO-PO invoices as the ERP system will not permit an invoice to be created without a vendor ID. Line pairing will not be carried out at the time of document export. If activated, the VAT registration compliance check will not be carried out. No vendor details will be exported.

With the exception of **PO VENDOR IS NOT INVOICE VENDOR**, **NON VAT COMPLIANT**, **THIRD PARTY FREIGHT** and **STOCKINVOICE**, line pairing will not be carried out during document export if an invalid reason is selected.

The *IVR section* of the project configuration allow an administrator to change the text, rule and export code associated with an invalid reason as well as add new invalid reasons based on an existing invalid reason rule. The invalid reason rules available are listed in the table above.

3.1.29 Invalid Reason Code

The Invalid Reason Code is the value that the system assigns to a selected invalid reason for the purposes of document export, so that a downstream workflow or ERP system can act upon that code and behave accordingly.

The code against each invalid reason can be set in the *IVR section* of the project configuration.

3.1.30 Employee ID

The Employee ID field represents the identification number or user name of an employee found on the document.

The can be used in a downstream workflow to route the document to relevant person within the customer's organization, for example, for invoice coding and approval.

The field can also be used to detect an employee, a department, or even a ship-to address on the document in order to help determine the cost object against which a non-PO invoice should be posted. In conjunction with the *Automatic General Ledger Account Coding* feature, this enables the complete coding of non-PO invoices.

The field is determined using the Associative Search Engine pointing to a CSV or database extract of the master employee data as specified in the *ASA section* of the project configuration.

3.1.31 Employee Name

The Employee Name is set by the results of the AP Packaged Project's associative employee search described above.

3.1.32 Line Item Detail

The AP Packaged Project will attempt to capture the following information at line item level:

Line Item Field	Mandatory	Description

Line Item Field	Mandatory	Description
PO	No	Purchase order number to which the invoice line item belongs. This is populated automatically by the line pairing routine should the invoice line be successfully paired to a line on the purchase order.
Line	No	Purchase order line item number to which the invoice line item relates. This is populated automatically by the line pairing routine should the invoice line be successfully paired to a line on the purchase order.
Material No	No	Material number associated with the invoice line item. If the material number captured is the material number from the point of view of the client, then this will be used to facilitate the line pairing function. For line item detail captured at the generic level, valid formats for the material number can be specified in the *MAT section* of the project configuration to assist the system is selecting the correct material number if possible.
GL Account	No	General Ledger Code to which the line item should be booked. This can be entered manually, or it can be determined automatically through the automatic general ledger account coding feature.
Description	No	Description of the invoice line item.
Quantity	Yes	Quantity being invoiced in the invoice unit of measure.
UOM	No	Unit of measure in which the invoice quantity is expressed.
Unit Price	Yes	Unit price quoted for the invoice line item.
Price Unit	No	The number of units for which the unit price is quoted. Example: For a line item where 5,000 units are invoiced at 100 dollars per 1,000 units, the line item total will be 500 dollars. In this case, the price unit is 1,000. This value will default to **1**.
Discount	No	Discount given by the vendor against the quoted unit price. This field can represent either a discount expressed as a percentage, or a hard amount to be subtracted from the quoted unit price. This field needs to be populated if the unit price captured is not the net unit price. During the line pairing operation, the unit price is compared against the purchase order unit price net of any discounts.
Total	Yes	Total value of the invoice line before tax.
Category	No	Miscellaneous charge category applied to the invoice line item based on the extracted line item description. This is set by the system automatically based on the settings specified in the *MSC section* of the project configuration. It cannot be changed by the Verifier user.
VAT Rate	No	Tax percentage rate applied to the invoice line item. If a valid value is captured, this is used by the system in the tax determination routine for countries that do not use tax. During document export, tax rates at line item level are cleaned up (e.g. if the total invoice tax amount is zero, then a rate of zero will be set for every line item; if the value captured is not between 0 and 100, it will be blanked out). Additionally, if automatic tax determination is activated, and tax codes are to be determined via a database lookup, the system will remove any tax rates that do not correspond to valid percentages listed in the table for the country in which tax applies. In order to increase extraction of tax rates, valid rates must be specified

Line Item Field	Mandatory	Description
		against the primary and secondary rate parameters in the *TAX section* of the project configuration. The system does not have the ability to extract custom vendor tax rate codes at line item level, and then subsequently perform a conversion to an actual percentage rate based on a legend the vendor specifies elsewhere on the invoice.
VAT Amount	No	VAT amount applied to the invoice line item. This column is also available for a user to enter line level ICMS tax amounts, which are mandatory for multi-line Brazilian Note Fiscal invoices where ICMS tax is being charged and no corresponding line level tax rates have been captured.

The project configuration options in the *TAB section* control whether:

1. Line items are required for any document.
2. Line items are required for NO-PO documents.
3. Line items are required for credit memos.
4. Line items are required for invoices relating to a service purchase order.
5. Only the line item total is required for invoices relating to a service purchase order.
6. Line items are required if the purchase order has not been released.
7. Line items are required for the MIRA scenario, which is when there is a one-to-one relationship between invoice and purchase order (i.e. the total value of the invoice matches either the total value of the purchase order, or the total value of all goods receipts against the purchase order that have not yet been invoiced).
8. Line items are required if either the **VENDOR NOT FOUND** or **MISSING/INVALID PO** invalid reasons have been selected by the user in Verifier.
9. Line items are required if the vendor is a utility vendor.

If line items are not required, or if an appropriate invalid reason is set, all of the line item table will be set to valid irrespective of content.

Each line item within the table is subject to the following validation formula:

*Line Total = (Quantity * ((Unit Price – Discount) / Price Unit)*

The discount can either be a hard value that is subtracted from the unit price, or as a percentage discount from the unit price.

The tolerance for the above calculations can be set in the *TOL section* of the project configuration. If the invoice relates to a service purchase order, then the above check is skipped if only the line total column is required.

3.1.33 Vendor VAT Registration Number and Bill-to VAT Registration Number

The Vendor VAT Registration Number and Bill-to VAT Registration Number are available to satisfy a European legal/fiscal compliance ruling, which states that, if value added tax is to be charged, it is incumbent on the vendor to state their VAT registration number and the VAT registration number of the bill-to party on the invoice.

The AP Packaged Project is able to carry out this compliance check automatically if VAT compliance checking is activated in the *TAX section* of the project configuration.

If activated, the system will look for the appropriate VAT registration numbers on the document, and any values found will be extracted into their corresponding fields. A valid vendor and company code must be present for this to occur.

If one or both VAT registration numbers cannot be found, tax is being charged, and both the vendor and company code are in EU member states, the document will be presented to a user in the Verifier application for them to key in the missing data. For the data to be accepted, both sets of VAT registration numbers must have the same ISO-code country prefix.

To enable automatic extraction, the VAT registration number of the vendor must be mapped in the *SRC section* of the project configuration; additionally, the bill-to company VAT registration number must be available via the company code validation in the *CCO section* of the project configuration.

The VAT registration number compliance check can be enabled or disabled on a company code by company code basis. It is also possible to configure the system to require the vendor VAT registration number only. VAT registration number checking is also supported for cross-border EU transactions where the VAT is zero-rated.

3.1.34 ICMS Tax Amount

ICMS Tax is a form of sales tax applied to material items in Brazil. It appears on a Brazilian Nota Fiscal invoice as a standalone tax value that cannot be validated in the same way as regular sales tax because the line item amounts on the invoice are already inclusive of this tax.

The AP Packaged Project will capture the total ICMS tax amount in the ICMS field. The regular AmountTax field is used to capture the IPI tax amount.

A document will stop in Verifier if the system believes it to be a Brazilian Nota Fiscal invoice referencing ICMS tax, yet no ICMS tax amount has been read. The user must then double-check whether ICMS tax was, in fact, present on the invoice.

A captured ICMS tax value, or one entered by the user manually, is validated mathematically by the application under the following circumstances:

1. More than one line item has been captured from the invoice.
2. The ICMS tax value is greater than zero.

If both of these conditions hold true, then the ICMS tax value must equal the sum of the values captured in the VAT Amount column in the table of line items. If no VAT amount at line item level has been extracted, then the system will try and use a captured VAT rate to determine what the VAT amount would have been. If this cannot be done, then the document will stop in Verifier for a user either to correct the ICMS tax amount, or to enter the line level ICMS tax amounts in the VAT amount column.

At time of export, the ICMS tax amount will be added on to the total invoice tax value, but is still available separately in its own export field. If line items are relevant for export, the line level unit prices and totals will be outputted exclusive of ICMS tax. During line pairing, the system assumes that pricing at the purchase order line item level is expressed exclusive of ICMS tax.

Usage of the ICMS tax amount must be activated in the *TAX section* of the project configuration.

3.1.35 Delivery Note

This field is used to capture the vendor delivery note number if it is stated on the invoice. By default, the extraction of this field is deactivated. Any extracted delivery note numbers are formatted in accordance with the formatting settings for the invoice number in the *NUM section* of the project configuration.

Refer to *Appendix I: Activating Optional Project Fields*.

3.1.36 Due Date

This field is used to capture the due date for payment. By default, this field is deactivated in the project, but can be enabled if required. Any extracted value will be formatted according to the settings in the *DAT section* of the project configuration.

Refer to *Appendix I: Activating Optional Project Fields*.

3.1.37 Delivery Date

This field is used to capture the delivery date for the goods/services stated on the invoice. By default, this field is deactivated in the project, but can be enabled if required. Any extracted value will be formatted according to the settings in the *DAT section* of the project configuration.

Refer to *Appendix I: Activating Optional Project Fields*.

3.1.38 Harmonized Sales Tax (HST)

Harmonized Sales Tax is a type of sales tax adopted by many Canadian states that combines the traditional Goods & Service Tax (GST) and Provincial Sales Tax (PST) into one single tax amount.

It can happen, however, that HST is stated on an invoice on top of GST and PST as a tax in its own right, which means that the system needs to be able to capture all three tax components in order for the invoice to pass.

In previous versions of the AP Packaged Project ,the AmountTax field was used to capture HST and the system will carry on behaving in this way unless parameter **BreakOutHSTForCanada** is set to **YES** in the *TAX section* of the project configuration. If it is set to **YES**, the AmountTax field will only be used to read GST. HST will be captured in the new HST field. If set to **NO**, HST will continue to be captured in the AmountTax field (which will lead to a document stopping in Verifier if both GST and HST are present on the document), and the content of the HST field will always be zero.

During the mathematical validation of the invoice amounts, the values of the HST, PST and regular tax field are added together in the background to form the total tax liability of the invoice. At time of document export, the tax amount exported is the sum of the regular tax field, the HST and the PST fields, although the HST component is available separately.

3.2 Solution Features

3.2.1 Line Pairing

The Line Pairing feature leverages WebCenter Forms Recognition's unique search technologies in order to reconcile the invoice line items with the line items on the purchase order.

This is a critical operation for creating a complete purchase order related invoice in the downstream ERP system, as ERP systems requires a purchase order line item number for each invoice line entered. The client's purchase order line item number does not habitually appear on

vendor invoices, and, when they do, they are not always stated correctly. The Line Pairing feature is able to overcome this and derive the correct purchase order line item number automatically through comparing the extract invoice line item data with what is available on the purchase order.

The WebCenter Forms Recognition differentiator is that the system employs the product's fuzzy search technologies to perform the above down to the line item description level, which delivers an industry-leading success rate.

Without this feature, although the document may pass straight through the Verifier application without requiring data correction, the document will always need to stop in the ERP system for manual completion. Often, this manual completion of the line item data can prove extremely time-consuming, especially when dealing with large purchase order numbers and a large number of invoice lines.

For example, if the purchase order contains 300 line items, and an invoice referencing this purchase order has 60 line items, the user would need to pick the relevant 60 lines from a list of 300 lines.

Hence, true *straight-through* processing for purchase order related invoices is rendered impossible unless line pairing is deployed.

In addition to this, the Line Pairing feature can also:

1. Perform checks to ensure that the invoice quantity is being booked in the correct unit of measure and convert to the purchase order unit of measure if required.
2. Reconcile the invoice data to blanket and service purchase orders within the ERP system.
3. Handle the posting of invoice miscellaneous charges (For example, freight, customs charges etc.) in accordance with client business rules.
4. Handle the same material appearing on the purchase order more than once.
5. Handle multiple purchase orders appearing on a single invoice.
6. Process third party freight invoices against a purchase order created for a different vendor.

To read the purchase order and goods receipt data required for line pairing, the AP Packaged Project can be pointed to a purchase order database, a flat file extract of purchase order line item data, or it can connect to an ERP.

Note: The flat file extract option is intended for demonstrational purposes only, and should not be used in a live production system.

The Line Pairing feature operates differently depending upon whether the invoice purchase order is for materials or services.

Line pairing will not be carried out if any of the following conditions are true:

1. Line pairing is deactivated in the project configuration.
2. Line item extraction is deactivated in the project configuration.
3. The invoice type is **NO-PO**.
4. The document type is **CREDIT**, and line item extraction is deactivated for credit memos.
5. The PO type is **SERVICE** and line pairing is deactivated for service PO types.

6. The vendor is a utility vendor and line item extraction is deactivated for utility vendors.
7. The Verifier user has selected an invalid reason of **VENDOR NOT FOUND**, and line item extraction is deactivated for that invalid reason.
8. The Verifier user has selected an invalid reason of **MISSING/INVALID PO**, **MISSING/INVALID VENDOR AND PO** or **INVOICE AMOUNTS DO NOT ADD UP**.
9. The purchase order has not been released, and line item extraction is deactivated for unreleased purchase orders.
10. All lines on the purchase order have been fully invoiced, and the system is configured to ignore completed purchase order lines.
11. A purchase order to be used for line pairing against has a duplicate record in the purchase order header database and duplicates are not allowed.

3.2.1.1 Line Pairing for Material Invoices

If the invoice relates to a material purchase order, the system undertakes these steps before a line item is paired:

1. Identify the corresponding purchase order line item.
2. Convert the invoice quantity to the order unit of measure specified on the purchase order line.

During step 1, the system looks at all the purchase order line items and, based upon the quantity, unit price, total, material number and description read from the invoice for each line item, the system will perform a fuzzy search against the purchase order data in order to identify the line that best fits the invoice details.

If a single line item is found within the tolerances specified in the *LPR section* of the project configuration, with a sufficient distance from the next best possibility, then the system selects this purchase order line.

In the second step, the system will check to see whether the purchase order line item identified in step 1 has been configured in the ERP system to require a specific goods receipt (i.e. the PO line is set for goods receipt based invoice verification). If a specific goods receipt is required for invoice entry, then the system will look at all of the available goods receipts for that purchase order line, and it will select a goods receipt (or combination of goods receipts) that reflect the amounts and delivery note details supplied on the invoice.

If an appropriate goods receipt cannot be determined, then line pairing will fail. If the purchase order line does not require a specific goods receipt to be specified when creating an invoice, then this check is skipped.

The final step is to ensure that the quantity extracted on the invoice is expressed in the same unit of measure as the purchase order line, and perform a conversion if necessary. This check can be disabled if required.

The mapping between an extracted unit of measure and the ERP ISO-code for the same unit of measure is configured in the *UOM section* of the project configuration.

If all three steps are successful, the invoice line item is successfully paired. This operation is subsequently repeated for all line items on the invoice.

3.2.1.2 Line Pairing for Service Invoices

The AP Packaged Project provides functionality for the handling of service invoices. Typically, for such invoices, line item detail is not extracted from the document as the line item breakdown provided by the vendor scarcely mirrors the manner in which a service purchase order is raised.

For example, a vendor providing consulting services may provide a complete breakdown of all time and costs spent on an engagement, each item of which constitutes an invoice line item, but purchasing departments are inclined to raise a single line blanket purchase order marked for **Consulting Services**, and the net invoice amount is simply booked against this single purchase order line.

The line pairing for service invoices adopts this approach and will only function if the purchase order comprises of a single line item.

It is common practice amongst many companies to reverse the quantity and unit price for a service line item on the purchase order as this:

1. Removes the need to pro-rata the quantity based on the invoice net total as a proportion of the overall purchase order line total at time of invoice entry.
2. Prevents the purchase order line being fully invoiced with the difference being posted to profit and loss.

For that reason, if the AP Packaged Project detects that the purchase order line has a unit price of **1**, the system will automatically book the value of the invoice into the quantity field with a unit price of **1**.

By default, any miscellaneous charges that appear on service invoices (e.g. freight) are considered part of the service value as a whole. If separate handling is required for such charges, then this should be specified in the *MSC section* of the project configuration by setting the **HandleMiscChargesForServices** parameter to **YES**.

3.2.1.3 Line Pairing for Third Party Freight Invoices

Within the AP Packaged Project, a third party freight invoice refers to a very specific business scenario whereby an invoice is received from a vendor billing for freight, yet that vendor legitimately quotes the purchase order number of another vendor (the material vendor), and it's against this material vendor's purchase order that the freight charge needs to be booked.

Freight invoices that do not fall into this category are handled as regular invoices.

Third party freight invoices are identified within the AP Packaged Project in two ways:

1. The identified vendor is not the material vendor for whom the purchase order was raised, but is the vendor set against a planned condition on any of the purchase order line items.
2. A user selects the **Third Party Freight** invalid reason from the drop-down within the Verifier application.

No line items are extracted from third party freight invoices. Instead, at time of line pairing, the system will book the net invoice amount according to the miscellaneous charge group settings assigned to third party freight vendors in the *MSC section* of the project configuration.

3.2.2 Automatic Tax Determination and Validation

The AP Packaged Project incorporates an automatic tax determination and validation feature in order to ensure that the document is correctly coded for tax prior to submission to the downstream ERP system.

This feature is in place so that a fully complete document can be created downstream; hence, manual rework in the ERP system is not required.

The system supports tax determination for countries with or without tax jurisdictions.

The determination of tax codes only applies to invoices that relate to purchase orders, and will only be carried out when a line item is successfully paired to its purchase order counterpart.

3.2.3 Data Export Options

The AP Packaged Project provides the following standard export options:

1. Export to database tables.
2. Standard extraction results file.
3. CSV file output.
4. XML file output.
5. TIFF file output.
6. Fully text-searchable PDF file output.
7. Integration to archive systems.

A user exit is provided for additional export requirements.

Export options can be enabled or disabled and configured via the *EXP section* of the project configuration.

The export event will fail if any of the following occur:

1. Late archiving is required, but the document cannot be archived.
2. The system is required to read the AP Project Tax Table for the purposes of tax determination, but the tax table could not be read.
3. The system is required to read the Miscellaneous Charges Account Assignment Table to code a general ledger entry, but the table could not be read.
4. The system is required to export the TIFF image to a designated directory, but the image cannot be written.
5. The system is required to export a PDF to a designated directory, but the document cannot be created.
6. A standard project results file is required to be created in a designated directory, but the file cannot be created.
7. A CSV output file is required to be created in a designated directory, but the file cannot be created.
8. Export is required to be written to a database, but the database insert/update is unsuccessful.
9. An XML file is required to be created in a designated directory, but the file cannot be created.
10. The system is required to do line pairing, but connectivity issues arise when trying to read purchase order data or service entry sheet data.
11. A custom export fails.

12. An unexpected error occurs.

Under such circumstances, the document will be set to state **750** (failed export), with an error message indicating the problem set against the Invoice Number field. Further detail is written into the standard log file for the RTS instance that performed the export.

The export will not fail if:

1. Line pairing was unsuccessful.
2. A document could not be successfully coded or validated for tax.
3. The update to the invoice number history database was unsuccessful.
4. The export data could not be written to the reporting database.
5. A database error occurred during the unit of measure conversion component of line pairing.

Unsuccessful SQL statements, such as those carried out for reporting, will be written into the standard log file for RTS instance that performed the export step. These can be executed manually at a future point in time.

If multiple export options are activated, export will terminate at the point at which the first export option fails. This will send the document to state **750** denoting an export failure. Upon retrying the export, only the export options that did not complete successfully upon the previous attempt(s) will be carried out. The system can be configured to repeat all export options (in the *EXP section* of the project configuration) irrespective of whether they had been completed beforehand.

3.2.4 Document Management System (DMS) Integration

The AP Packaged Project supports integration to document management systems in both the early and late archiving scenarios.

3.2.4.1 Early Archiving

Early archiving means that the image has already been archived prior to being processed through WebCenter Forms Recognition. In this scenario, the AP Packaged Project requires a copy of the archived image with the unique archive document ID embedded into the document filename.

Configuration options in the *IMP section* define whether this unique archive document ID constitutes the entire filename, or an underscore-separated component.

At time of document export, the archive document ID is passed downstream via the URN.

3.2.4.2 Late Archiving

Late archiving means that the image is to be archived after processing through the AP Packaged Project.

3.2.5 ERP System Integration

Integration to downstream ERP systems with the AP Packaged Project is possible via the following interfaces:

1. Flat file transfer.
2. Export to database staging tables.

3. Direct creation of invoice transactions in the host ERP solution.

The various export options can be activated in the *EXP section* of the project configuration.

3.2.6 Project Reporting

The AP Packaged Project can be configured to populate reporting tables in the Forms Recognition database to record processing metrics for the Runtime Service, the project itself and the individual documents being processed. For more information about the reporting features and how to enable them, refer to *Appendix L: Configuring AP Project Reporting*.

3.2.7 Automatic General Ledger Account Coding

The AP Packaged Project provides a feature by which non-purchase order related invoices can be coded automatically for general ledger entry, and the appropriate cost object assigned.

Both the general ledger coding and cost object determination are carried out using the project's unique search technologies which have the ability to reconcile free-text information on an invoice to structured data.

By virtue of the extracted line item description, the search technology is used to allot an appropriate general ledger code based on what code has been used for that item in the past; the cost object information is derived via contact names, department names and ship-to addresses that the vendor may have included on the invoice document.

These two items come together to provide a complete set of coding strings for the invoice in order that they can be submitted to the downstream ERP system.

Note: The implementation of the auto GL-coding feature requires customization, typically within *UserExitPostLinePairing*.

3.2.8 Duplicate Invoice Number Checking

The AP Packaged Project provides the ability to identify a duplicate invoice, by performing a lookup against a configured data source; typically the customer's ERP database. This check is performed as part of the document validation procedure, which means it will be performed both on the RTS instance after data extraction, and also within the Verifier application (and Web Verifier). Therefore, both extracted Invoice Number values, and manually entered Invoice Number values will be subject to this check for duplicates.

Different organizations may have varying definitions of what constitutes a 'duplicate' invoice. Typically, a duplicate invoice is defines as one coming from the same vendor (including the vendor site) with the same invoice number and issued to the same company. The AP Project provides configuration options to define which of these values should be used to define a duplicate invoice for a particular implementation.

Where a duplicate invoice is identified, the document will be prevented from being exported, and will require manual verification so the appropriate action can be taken. The system will not allow an invoice with a duplicate invoice number to be released and exported unless the configuration allows the Verifier user to forcefully validate and release the document with the duplicate invoice number in place.

This feature can be enabled and configured through the *WFR section* of the project configuration file.

3.2.9 Credit Note Export: Negative Quantity and Totals

Some ERP systems, such as Oracle E-Business Suite, require that when a credit note is inserted into the system through its automatic entry interface the header-level total and the quantity and total at line-level are expressed as negative values.

The AP Packaged Project exports numeric values as positive numbers by default, but this feature allows the project to be configured to always export the quantity and total amounts as their negative values when the document is a credit note. If enabled, this will occur for all credit notes, irrespective of whether the original document expressed those values as positive or negative values.

This feature can be enabled through the *WFR section* of the project configuration.

3.2.10 Metadata Pass-through

The AP Packaged Project provides the ability to parse the image filename, and write component values to the CSV and/or XML files that are created during export. This is in addition to the basic import functionality described in the *IMP section* of the project configuration.

Unlike the basic import function, where image filename component values are assigned to fields within the Invoices class, this Metadata Pass-through feature does not require fields to be created to hold the component values.

Both the basic import function and this Metadata Pass-through feature can be enabled within the same project, and work independently of each other.

This feature requires that the image filename use the underscore character as a separator to delimit the components. For example:

Assume the image filename: **007_00000011_204_05-26-2011_EMEA.tif**

The image filename in this example has five components:

- 007
- 00000011
- 204
- 05-26-2011
- EMEA

You can configure up to 99 components through the project configuration file, and for each configured component, you can specify whether it should be written to the CSV file, the XML file or both files at export. An example configuration could be:

```
CXP_VL_01_FilenameComponent=1
CXP_VL_01_CSVFieldID=<%ZXX>
CXP_VL_01_XMLHeaderFieldName=imageComponent1

CXP_VL_02_FilenameComponent=2
CXP_VL_02_CSVFieldID=<%ZYY>
CXP_VL_02_XMLHeaderFieldName=

CXP_VL_03_FilenameComponent=3
CXP_VL_03_CSVFieldID=
CXP_VL_03_XMLHeaderFieldName=imageComponent3
```

In the example configuration given above, component 1 will be written to both the CSV file and the XML file, component 2 will be written only to the CSV file, and component 3 will be written to only the XML file.

This feature can be enabled and configured through the *WFR section* of the project configuration.

3.2.11 PO Number Removal for Non-PO Invoices

By default with the AP Packaged Project, if a document is manually processed in the Verifier application, and the user changes the Invoice Type from **PO** to **NO-PO**, any existing value in the PO Number field will remain intact.

For some ERP integrations, this situation, where a non-PO invoice has a PO number value, can cause the invoice to be created in the ERP system erroneously as a PO-based invoice.

If enabled, the PO Number Removal feature will ensure that during document validation any PO Number value is removed if the Invoice Type is set as **NO-PO**.

This feature can be enabled through the *WFR section* of the project configuration file.

3.2.12 Separator Page Detection for Supporting Documents

Invoices are commonly received with supporting documentation attached. While customers typically want to retain these as a single document in the downstream content repository after Forms Recognition has processed them, the supporting pages should not be considered during extraction.

While WebCenter Forms Recognition does allow for the first *n* pages and /or the last *n* pages of a document to be OCR'd, this functionality cannot be effectively used in this situation, where invoices are a variable number of pages.

Instead, all pages of the document should be OCR'd as normal, then this Separator Page Detection feature allows the project administrator to configure one or more phrases, which, if found on the document, should be considered as a separator. In this case, all OCR text following this separator will be discarded from the workdoc, and will not be considered during the extraction step of the Forms Recognition workflow. This feature can also be configured to remove the page containing the separator phrase(s) from the exported image if required.

Note: If separator page removal is enabled, additional settings must be configured in the *EXP section* and the *CSV section* of the project configuration to ensure that the modified image file is exported from Forms Recognition instead of the original image that was imported. These settings are described in *Section 4.1.33: WFR Section*.

This feature can be enabled and configured through the *WFR section* of the project configuration.

3.2.13 Export Custom Unit of Measure Value to XML

There may be situations where a customer may wish to export a line item *Unit of Measure* value that differs from that on the purchase order line. For example, the unit of measure value may be expressed on the invoice, and extracted by the AP Packaged Project, as **EACH**, but the output requirement may be that the exported value is written as **Each** in the XML output file.

In some scenarios the required output may be entirely different, for example an extracted value of **EACH** may be required to be written to the XML as **EA**.

This feature allows a project administrator to specify whether one or more units of measure should be exported as a different value to the one that was extracted, and to configure the value that should be written to the XML in each case.

This feature can be enabled and configured through the *WFR section* of the project configuration.

3.2.14 Force Validation of Documents Using Custom Invalid Reasons

Standard functionality of the Verifier application provides the user with a couple of options for handling invalid documents, or documents that cannot be successfully validated and released for export:

1. The user can set the document to an exception state and (optionally) move it to a separate batch for processing later.
2. The user can manually reclassify the document to the *Void* class.

While these options are sufficient in the majority of cases, there may be a business requirement to allow an invalid document to pass verification and be released for export. This is particularly true where the downstream process will be able to identify this type of document and process it accordingly. Again, standard functionality within Verifier will often allow users to set an appropriate *Invalid Reason* to allow documents that could otherwise not be validated to continue through to export.

However, the *Invalid Reason* approach is limited because each invalid reason is linked to a rule, and none of the available rules make all fields in a document valid, so there may still be some fields that require user entry or correction. In the case of a bad image or a document that is not an invoice, such manual verification may not be appropriate, or even possible.

This feature allows an administrator to configure the project so that the Verifier user can forcibly validate a document, and allow it to be released to export, by selecting one of a defined list of invalid reasons.

This feature can be enabled and configured through the *WFR section* of the project configuration.

3.2.15 Format Line Items in XML for the E-Business Suite Open Interface

In some cases where E-Business Suite is the downstream ERP system, the XML output from the AP Packaged Project may not meet the specific validation criteria of the EBS open interface. This may result in the import failing or invoice lines being imported with EBS holds applied unnecessarily.

For example, in a scenario where the **PURCHASE_BASIS** EBS field is configured to determine a service PO, and the value of that field is **SERVICES**, that value is exported to the XML file as the line type, e.g.:

```
<lineType>SERVICES</lineType>
```

However, **SERVICES** is not a valid line type value for the EBS open interface, and this value should actually be exported in the XML file as **ITEM**.

Additionally, for certain line types, such as **FREIGHT** and **MISCELLANEOUS**, the EBS open interface may expect the quantity, unit price and unit of measure to be zero or empty. Finally, for invoices where the PO type is **SERVICE** the EBS open interface may require the quantity and unit price values to be transposed in the XML file.

This feature allows the project administrator to configure how the line item data is written to the XML file in all of the examples described above, to ensure that the resulting file content will not cause any unnecessary rejections or failures during import into E-Business Suite through the open interface.

This feature can be enabled and configured through the *WFR section* of the project configuration.

4 Configuration Settings

The AP Packaged Project can be configured via the project configuration (.ini) file, which resides in the same directory as the project (.sdp) file. The configuration file contains a multitude of settings that control the way in which the project behaves. This section describes the configuration settings available, and how they may be used.

4.1 Configuration File Settings

The project configuration is sub-divided into various sections that control different aspects of the project's behavior.

Each parameter in the project configuration is made up of the following nomenclature:

XXX_YY_DDDDD=ZZZ

or:

XXX_YY_NN_DDDDD=ZZZ

XXX = the project configuration section ID (e.g. **REP**, **GRL**, **ITY**, **EXP**, etc.)

YY = the type of setting, where **VL** denotes a value or list of values, and **OP** denotes an on/off switch and should be set either to **YES** or **NO**

NN = optional project configuration group ID used to tie multiple individual settings together to form a settings group. This is similar to a database table where *XXX* is table name, *NN* represents the unique table row and *DDDDD* represents the unique table column name.

DDDDD = the parameter name, which may be more or less than 5 characters

ZZZ = the parameter setting, which can be completed by the individual configuring the project and may be more or less than 3 characters

Only *ZZZ* values should ever be changed in the file, though additional *NN* settings groups may also be added as appropriate.

4.1.1 GRL Section

This section contains global settings for the project that are used for the purposes of reporting to the AP Project Reporting tables in the Forms Recognition database.

Parameter	Type	Description
ProjectName	Freetext	The name of the project.
Version	Number	Version number of the project implementation.
ClientName	Freetext	The name of the client/customer.

4.1.2 IMP Section

This section contains settings relating to document import, specifically the mapping of values contained within the image filename to fields in the AP Packaged Project. This provides a simple means to pass metadata to the project from an upstream system, such as WebCenter Enterprise Capture.

Note: An underscore character must be used to separate components within the image filename.

For example, assume an image filename of **002_00000006_204_20141208.tif** and the following settings:

```
IMP_VL_URN=COMPONENT2
IMP_VL_BatchName=COMPONENT1
IMP_VL_ScanDate=COMPONENT4
IMP_VL_CompanyCode=COMPONENT3

IMP_VL_DateFormat=YYYYMMDD
```

In this example, the value **20141208** will be used to populate the ScanDate field for this document. Notice that the date format in this example is specified as **YYYYMMDD** so the scan date value will be interpreted as 8th December 2014.

Parameter	Type	Description
URN	Freetext	Document unique reference number.
BatchName	Freetext	Document batch name.
ScanDate	Freetext	Document scan date.
PriorityFlag	Freetext	Document priority flag.
InvoiceType	Freetext	Document invoice type.
DestinationArchive	Freetext	Document destination archive.
CompanyCode	Freetext	Document company code.
InputSource	Freetext	Document input source. (For example, **SCAN**, **EDI**, **EMAIL**, etc.)
PriorityFlagYes	Freetext	Value that denotes a positive setting for the priority flag.
DateFormat	Freetext	Format of a date contained within the document filename. Options are: ▪ DDMMYYYY ▪ MMDDYYYY ▪ YYYYMMDD

4.1.3 REP Section

This section contains the configuration settings relating to the AP Packaged Project reporting features.

Parameter	Type	Description
ConnectToReportingDB	YES/NO	Flag to set whether the project will write out reporting data or not.
SQLConnectionGroup	NN	The SQL connection group specifying the reporting database connection string as set in the *SQL section*. If no connection group is specified, the system will use group **01**.
ReportingInDesigner	YES/NO	Flag to indicate whether documents processed or analyzed in the Designer application should have the results written to the reporting database.
StartNewRecordFor ImportedDocument	YES/NO	If this is set to **YES**, a new reporting record will be created for each document imported into Designer, removing any old ones for the same document key. If this is set to **NO**, WebCenter Forms Recognition will only write to the reporting database if an entry exists for the same document key. This can be used in the event that the reporting trail begins at the scan station.
ReportingDBDocument	Freetext	Name of the document header table in the reporting database.

Parameter	Type	Description
Table		
ReportingDBFieldTable	Freetext	Name of the document field table in the reporting database.
ReportingDBHistory Table	Freetext	Name of the document history table in the reporting database.
ReportingDBImage Table	Freetext	Name of the document image table in the reporting database.
StoreImageInReporting Tables	YES/NO	Indicates whether the document image should be stored in a binary type field in the reporting database.
ReportingKey	Freetext	Contains the component to be used as the database table key for the document record. Valid values are: ▪ URN ▪ *(blank)* If this is set to **URN** then the unique reference number will be used in the reporting tables as the document key. In this case, the IMP_VL_URN setting must be correctly mapped in the *IMP section*. If left blank, the key will be set to the image filename (minus the file extension).
ArchiveURL	Freetext	Contains the mask for the URL associated with a document link. **XXXXX** should denote the part of the URL that should be substituted with the unique document ID from the point of view of the archiving system to form a valid URL, which will retrieve the document.

4.1.4 SQL Section

This section contains the SQL connection strings that are used by the AP Packaged Project.

The solution works with OLE DB compliant data sources. It requires that the appropriate 32-bit OLE DB provider be correctly installed on the computer where the AP Packaged Project is being executed.

Parameter	Type	Description
NN_ConnectionString	Freetext	Connection string for SQL group NN.

4.1.5 ASA Section

This section contains settings that control the ASSA pools used for the vendor, employee and company code lookups in the AP Packaged Project.

If an ASSA field is not required, it can be removed from the project configuration file as long as the ASSA setting is also disabled in the project file. Refer to *Appendix F: Deactivating ASSA Fields*.

Parameter	Type	Description
Class	Freetext	Name of the project class on which the field was created.
Fieldname	Freetext	Technical name of the project field.
AlphaNum	YES/NO	It set to **YES** this indicates that the key field for the pool record is alphanumeric. If set to **NO** the field is assumed to be numeric. This must be set correctly in order to generate the pool correctly
PoolRelative	YES/NO	Indicates whether the location of the pool directory is relative to the project file.
PoolPath	Freetext	UNC path to the pool directory if it is not relative to the project file.

Parameter	Type	Description
PoolDirectory	Freetext	Name of the pool directory.
PoolName	Freetext	Name of the pool.
FileRelative	YES/NO	Indicates whether the location of the pool import CSV file is relative to the project (.sdp) file.
ImportPathFilename	Freetext	UNC path to the pool import CSV file if it is not relative to the project file.
ImportFilename	Freetext	Name of the pool CSV import file.
ImportODBCDSN	Freetext	Name of the user DSN for the ODBC pool import.
ImportODBCSelect	Freetext	Select statement used to create the pool.
ImportODBCUser	Freetext	User ID used to connect to the ODBC data source. This can be left blank and specified in the project file through the Designer application if security requires it.
ImportODBCPWD	Freetext	User password to access the ODBC data source. This can be left blank and specified in the project file through the Designer application if security requires it.
AutoImportOption	Freetext	Indicates the source from which the pool should be created via the WebCenter Forms Recognition Runtime server. Valid settings are: ▪ FILE ▪ NONE ▪ ODBC If set to **NONE** the pool will not be updated automatically by the RTS.
FirstPageOnly	YES/NO	Flag to indicate whether only the OCR text on the first page of the document should be used to determine the field result.
PageZoneALeft	0-100	Zone A left search %
PageZoneAWidth	0-100	Zone A width search %
PageZoneATop	0-100	Zone A top search %
PageZoneAHeight	0-100	Zone A height search %
PageZoneBLeft	0-100	Zone B left search %
PageZoneBWidth	0-100	Zone B width search %
PageZoneBTop	0-100	Zone B top search %
PageZoneBHeight	0-100	Zone B height search %

4.1.6 IMG Section

This section contains a single setting related to image pre-processing prior to OCR.

Parameter	Type	Description
ConvertTo300	YES/NO	Flag to indicate whether WebCenter Forms Recognition should increase the dpi to 300 for images prior to OCR.

4.1.7 PON Section

This section controls the extraction and validation of purchase order numbers, as well as how service purchase orders are to be identified.

Parameter	Type	Description
NN_Format	Freetext	Format string NN for a valid purchase order number. Numerous purchase order number formats can be entered using incremental values (e.g.

Parameter	Type	Description
		01_Format, 02_Format, 03_Format). **#** is a wildcard character representing any number; **@** represents any alpha character For example, **45########** would denote a possible purchase order number as a ten-digit number beginning with **45**. **@@######** would denote a possible purchase order number as being two alpha characters followed by five digits. Oracle recommends defining these strings as tightly as possible to optimize the system being able to distinguish the correct value.
NN_Ignore	Freetext	List of characters that may appear in a purchase order number in any position (e.g. a hyphen or a period/full stop) that the system should be tolerant of in the corresponding format string specified. This list does not need to be comma separated.
MaxWordCount	Freetext	This value should be set to the maximum number of OCR words that are permitted to form a candidate for the purchase order number. The recommendation is to set this to **3**. This would allow: **45 0000 0020** or **12345 – OP** to be recognized as possible purchase order numbers, where each value separated by a space is recognized as an individual OCR word. A hyphen (-) will always be considered by the system to be a separate OCR word, irrespective of the geometric distance to neighboring values on either side.
SetCompanyCodeFromPO	YES/NO	If set to **YES**, the system will overwrite any existing content in the company code field with the company code derived from the purchase order.
ValidateFromDB	YES/NO	Flag to denote whether the extracted purchase order number should be validated against a database table. If this validation is activated, the extracted purchase order number must exist in the database table and the vendor identified on the invoice must be the vendor on the purchase order. In the case of Oracle implementation, the invoice site ID address does not have to be the same as the PO order-from vendor site ID. In all other cases, the purchase order number field will be set to invalid.
POKeyIncludesCompany Code	YES/NO	This value should be set to **YES** if the unique key to identify a single purchase order in the PO header database lookup table consists of the purchase order number and the company code in tandem. This should be set to **NO** for JD Edwards implementations.
UseStoredProcedure	YES/NO	This value should be set to **YES** if the PO header details are to be retrieved from a database using a stored procedure.
StoredProcedureName	Freetext	This Is the technical name of the stored procedure to be used to retrieve the PO header details.
StoredProcedureParameters	Freetext	This is a comma-separated list of parameters relevant to calling the stored procedure. The parameters listed must correspond to entries maintained in the *SPC section* of the project configuration file.
SQLConnectionGroup	NN	SQL connection group specifying the purchase order database connection string as set in the *SQL section*. If no connection group is specified, the system will use group **01**.
DBTableName	Freetext	Name of the database table containing the purchase order header information. This setting is mandatory for database validation where a stored procedure is not used.
DBPO	Freetext	Name of the database table field holding the purchase order number.

Parameter	Type	Description
DBVendorID	Freetext	Name of the database table field holding the vendor ID for a given purchase order. This setting is mandatory for database validation. This must always be set to the internal vendor ID that the ERP system uses.
DBSiteID	Freetext	Name of the database table field holding the site ID for a given purchase order. This must be mapped when working with ERP systems that use a vendor ID and a site ID to identify a unique vendor address (e.g. Oracle Financials). The column must not be mapped under any other circumstances.
DBCurrency	Freetext	Name of the database table column holding the purchase order document currency. This is not mandatory.
DBCompanyCode	Freetext	Name of the database table column holding the company ID for which the purchase order was created. This is only mandatory for JD Edwards purchase orders, or where the **POKeyIncludesCompanyCode** parameter is set to **YES**.
DBStatus	Freetext	Name of the database table column holding the status of the purchase order (i.e. is it released, closed, etc.) This is not mandatory.
DBDocType	Freetext	Name of the database table column holding the purchase order document type. This is only mandatory for JD Edwards purchase orders.
DBBusinessUnit	Freetext	Name of the database table column holding the purchasing business unit. This is only mandatory for PeopleSoft purchase orders.
LengthForLeadingZeros	Freetext	This is the length that the purchase order number field should be, including leading zeroes. For example, if this value is set to **10**, and **123456** is extracted as the purchase order number, or entered by a user into the purchase order number field, then the system will convert this value to **0000123456**.
JDEPO	YES/NO	If this value is set to **YES**, the system will look for and validate purchase order numbers based on the JD Edwards ERP system formats and data structures, i.e. a unique purchase order is identified by the combination of company code, purchase order number and purchase order type. In Verifier, the *PO Extension* field will become a mandatory field for PO-based invoices where the JD Edwards purchase order type should be entered.
JDEPOTypes	Freetext	Comma-separated list of valid JD Edwards purchase order types. If **JDEPO** is set to **YES**, purchase order numbers will not be extracted unless a valid purchase order type is also specified on the document, before or after the actual purchase order number itself. Typical JD Edwards purchase order types are **OP** and **UX**.
PeopleSoftPO	YES/NO	If this value is set to **YES**, the system will look for and validate purchase order numbers based upon the PeopleSoft ERP system formats and data structures, i.e. a unique purchase order number is identified by a combination of the purchase order number itself and the purchasing business unit. In Verifier, the *PO Extension* field will become a mandatory field for PO-based invoices where the purchasing business unit should be entered.
PeopleSoftBusinessUnits	Freetext	Comma-separated list of valid PeopleSoft purchasing business units. If **PeopleSoftPO** is set to **YES**, purchase order numbers will not be extracted

Parameter	Type	Description
		unless a valid purchasing business unit is also specified on the document, before or after the actual purchase order number itself.
ServicePOTypes	Freetext	Comma-separated list of purchase order document types that denote a service purchase order.
ServicePOItemCategories	Freetext	Comma-separated list of item categories/line types at the purchase order line item level that denote a service.
ServicePOPrefixes	Freetext	Comma-separated list of purchase order prefixes that exclusively identify a service purchase order
ServicePOUOMs	Freetext	Comma-separated list of units of measure at the purchase order line item level that exclusively identify a service purchase order.
StopERSPO	YES/NO	If set to **YES**, the system will stop a purchase order in Verifier if it is marked as being an ERS/self-billing purchase order.
StopERSPOInVerifier	YES/NO	If set to **YES**, the system will not permit a purchase order to pass Verifier if it is marked as being an ERS/self-billing purchase order.
SkipDuplicatePOCheck	YES/NO	If set to **NO**, the system will stop a document in Verifier if a database lookup is activated and the purchase order number exists more than once in the purchase order header table. The user may choose to accept the PO in Verifier, but line pairing will not be carried out. If set to **YES**, the system will not reject a purchase order if a duplicate record is found in the purchase order number header table (but instead will use the first one found), and line pairing will be carried out as usual. If the purchase order is keyed using a combination of the purchase order number and a company code/business unit/PO document type, which can be the case for implementations involving JD Edwards or Peoplesoft, then a duplicate will only be flagged if multiple records are found using all keys defined.

4.1.8 BTO Section

This section is used to specify format strings for possible bill-to names on the invoice, which will be extracted into the Bill-to Name field.

Parameter	Type	Description
AllowBlank	YES/NO	If set to **YES**, the system will not set to Bill-to Name field to invalid if a valid bill-to name has not been extracted on server side.
NN_Format	Freetext	Format string NN for a possible bill-to name. Numerous bill-to name formats can be entered using incremental values (e.g. 01_Format, 02_Format, 03_Format). **#** is a wildcard character representing any number; **@** represents any alpha character. The format string entered is used to help the system identify the correct bill-to name. For example, if **ACME** is specified as a possible bill-to name, the system will use this to help anchor the bill-to name on the invoice, but the field will be extracted as it appears on the document. Hence, if the bill-to name on the invoice were actually **ACME Inc. UK Office**, then this would be the value extracted.
NN_Ignore	Freetext	List of characters that may appear in a bill-to name in any position (e.g. a hyphen or a period/full stop) that the system should be tolerant of in the corresponding format string specified. This list does not need to be comma separated.

4.1.9 AMT Section

The settings in this section specify the format string and ignore characters for a valid amount used by the project.

Oracle does not recommend that this setting be changed from the delivered default which is:

```
AMT_VL_Format=#[2-10]
AMT_VL_Ignore=,.'-_$£€
```

More ignore characters may be added to cover a wider variety of currency symbols if appropriate for a particular implementation.

Parameter	Type	Description
Format	Freetext	Simple expression denoting the format of an amount.
Ignore	Freetext	Special characters that may appear in the amount.
ValidateTotalOnly	YES/NO	If set to **YES**, the system will validate the total captured based on the standard project field validation settings (i.e. system OCR confidence and candidate distance), and will not require this value to be in balance with other amount fields captured from the invoice. This setting can be set to **YES** for more basic invoice extraction projects where only the invoice total amount field is required.
SubtotalRequired	YES/NO	If set to **YES**, the invoice subtotal amount, which is usually not mandatory, will become a required field and must be populated with a numeric value that is in balance with the other amount values captured at header level, i.e. either of the calculations below hold true: *Invoice Total = Subtotal + Total Tax + Freight + Miscellaneous Charge – Discount – Withholding Tax* or: *Invoice Total = Subtotal + Total Tax – Discount – Withholding Tax* ***AND*** *Invoice Total = Total of Line Items + Total Tax + Freight + Miscellaneous Charge – Discount – Withholding Tax* This setting is ignored if **ValidateTotalOnly** is set to **YES**. It is not recommended to activate this parameter in projects where vendors do not always specify a subtotal on the invoice.

4.1.10 DTY Section

This section is used to specify words that may appear on an incoming document that would denote that the document is a credit memo.

Parameter	Type	Description
NN_Format	Freetext	Format string *NN* for a possible credit note indicator. Numerous credit note indicators can be entered using incremental values (e.g. 01_Format, 02_Format, 03_Format). **#** is a wildcard character representing any number; **@** represents any alpha character.
NN_Ignore	Freetext	List of characters that may appear in a credit note indicator in any position (e.g. a hyphen or a period/full stop) that the system should be tolerant of in the corresponding format string specified. This list does not need to be comma separated.
Distance	Freetext	This represents the 'fuzzy' factor the system uses when determining document types based on the entries above. It is a value between zero and one, where zero requires an exact match, and one accepts values that do not

Parameter	Type	Description
		match at all. The recommended value for this setting is **0.17**.
StopAllCredits	YES/NO	If this flag is set to **YES** then all documents that the system identifies as being a credit will stop in Verifier for manual review.
IgnoreNegativeTotal	YES/NO	If set to **YES**, the system will not use the presence of a negative invoice total (as indicated by a minus sign before or after the value) to determine whether the document type should be a credit memo, as opposed to an invoice.

4.1.11 ITY Section

This section includes configuration settings that determine the Invoice Type field in the AP Packaged Project, and whether it is set to **PO** or **NO-PO**.

Parameter	Type	Description
Default	Freetext	Specifies the default value for the field. Valid settings for this parameter are: ▪ PO ▪ NPO **PO** = purchase order related; **NPO** = non-purchase order invoice. If the invoice type exists in the image filename and has been mapped in the *IMP section*, then this overrides the default.
SetByVendor	YES/NO	Specifies whether the Invoice Type field should be set by the extracted vendor. For this to occur, the invoice type value must be mapped to a column in the vendor extract in the *SRC section*.
SetByVendorCCExceptions	Freetext	Comma-separated list of company codes that are exceptions to the **SetByVendor** parameter above. For example, if **SetByVendor** is set to **YES**, and **SetByVendorCCExceptions** is set to **1000,2000**, then the system will set the invoice type according to the vendor except when the invoice belongs to company code 1000 or 2000. If **SetByVendor** is set to **NO**, then the system will set the invoice type according to the vendor only if the invoice belongs to company code 1000 or 2000.
POValue	Freetext	Denotes the value held in either the document filename or in the vendor extract that represents a PO-based invoice.
NPOValue	Freetext	Denotes the value held either in the document filename or in the vendor extract that represents a non-PO invoice.
SetToPOIfPOFound	YES/NO	On the RTS, if this setting is set to **YES** and a purchase order number is found, the invoice type will be set to **PO** irrespective of any default or vendor specific settings.
SetToPOIfValidPOFound	YES/NO	On the RTS, if this setting is set to **YES** and an extracted purchase order is found to exist in a validation database or ERP system, the invoice type will be set to **PO** irrespective of any default or vendor specific settings.
SetToPOIfPOPopulated	YES/NO	If this parameter is set to **YES**, the system will always set the invoice type to **PO** if any input is present in the purchase order number field, whether captured by the RTS, or entered manually by a Verifier user.

4.1.12 NUM Section

This section contains the formatting and validation options available for the Invoice Number field.

Parameter	Type	Description

Parameter	Type	Description
RemoveAllSpecials	YES/NO	If set to **YES**, all special characters will be removed from the extracted invoice number.
KeepCertainSpecials	Freetext	Non comma-separated list of special characters that should be retained if the **RemoveAllSpecials** setting is set to **YES**.
RemoveSpecialsAtStartAndEnd	YES/NO	If set to **YES**, any special characters at the beginning and at the end of an extracted invoice number will be removed.
RemoveSpaces	YES/NO	If set to **YES**, any spaces that are present in the extracted Invoice Number value will be removed. If set to **NO**, any spaces in the extracted Invoice Number will be retained in the value that is exported. Where the extracted Invoice Number does not contain any spaces, this setting will have no effect.
RemoveLeadingZeros	YES/NO	If set to **YES**, any leading zeros will be removed from an extracted invoice number.
SkipForUtilityVendor	YES/NO	If set to **YES**, and the vendor is identified as being a utility vendor (if the column is mapped in the *SRC section*), then the Invoice Number field will be set to valid and the *Account Number* will be required to be populated instead.
UtilityAlias	Freetext	Specifies the positive value in the vendor extract that denotes a utility vendor.
AcceptTwoCharacters	YES/NO	If set to **YES**, the system will not automatically set a two character invoice number to invalid on server side. Otherwise, any invoice number extracted which is two characters or less in length will be marked as invalid by the system automatically, and the document will be sent to Verifier for a user to confirm the extracted value.
ValidateFromDB	YES/NO	Flag to denote whether the invoice number should be validated against previous invoice numbers from the same vendor in a database table.
SQLConnectionGroup	NN	SQL connection group specifying the invoice number history database connection string as set in the *SQL section*. If no connection group is specified, the system will use group **01**.
DBTableName	Freetext	Name of the invoice number history database table.
VendorID	Freetext	Column in the database table that holds the vendor number. This setting is mandatory.
RecID	Freetext	Column in the database table that holds the record ID. This setting is mandatory if the invoice number history table is to be updated upon export.
InvoiceNumber	Freetext	Column in the database table that holds the invoice number. This setting is mandatory.
DocumentType	Freetext	Column in the database table that represents the document type; either invoice or credit. This setting is mandatory.
InvoiceAlias	Freetext	Value in the document type column in the invoice number history table that denotes an invoice. This setting is mandatory.
CreditAlias	Freetext	Value in the document column in the invoice number history table that denotes a credit memo. This setting is mandatory.
MaxRecords	Freetext	Maximum number of records that should be considered when comparing an extracted invoice number to historic invoice numbers from the same vendor

Parameter	Type	Description
		in the database table. The default is set to **20** records.
NoOfHits	Freetext	Number of hits required to validate the extracted invoice number format with the invoice number history database table. The default is set to **2** records.
CorrectOCRMisreads	YES/NO	If set to **YES**, and database validation of the invoice number is activated, the system will look to repair two specific OCR issues with the extracted invoice number. If a **1** has been read in the extracted value, but the invoice history shows that this should be the letter **I** in as many records as set in the **NoOfHits** parameter, then the extracted value will be changed to an **I** automatically on server side only. This also applies to instances where a zero has been extracted, and the history shows that a letter **O** is expected. No other OCR issues are handled.
CheckSequencing	YES/NO	If this flag is set to **YES**, and the vendor supplies invoice numbers that are entirely numeric, then, when carrying out the invoice number format comparison against the vendor's prior history, the system will not consider prior invoice numbers if they are 100 ahead or more. For example, invoice number **1500** is read from an invoice. In the history table, invoice numbers **1598**, **1599**, **1600** and **1601** are present, **1600** and **1601** will not be considered valid matches as, although they match the four-numeric format, they are too far ahead in terms of the sequence. The sequence limit, in the above example set to **100**, is configurable via the **SequencingLimit** parameter.
SequencingLimit	Integer	This setting is used in tandem with the sequence check functionality described above. If no sequence is specified (i.e. the sequencing limit is set to zero), then the system will use the default sequencing limit of **100**.
UpdateDBAtExport	YES/NO	If set to **YES**, the system will update the invoice number history table with a record for the current document at the point of document export. This setting should be set to **NO** if the history table is not maintained and managed by WebCenter Forms Recognition. For example, a table or view in an ERP system's database.
DeliveryNoteRequired	YES/NO	If set to **YES**, the *Delivery Note* field will be made mandatory and the document will be sent to Verifier if no value is extracted. The Verifier user will be permitted to let the field pass blank.
RemoveAllSpecials	YES/NO	If set to **YES**, all special characters will be removed from the extracted invoice number.

4.1.13 DAT Section

The settings in this section control the formatting and validation of the invoice date.

Parameter	Type	Description
VerifierOutputFormat	Freetext	Defines that format in which dates will be displayed in Verifier. Valid settings for this parameter are: ▪ DDMMYYYY ▪ MMDDYYYY ▪ YYYYMMDD If set to **DDMMYYYY**, the date will be displayed in Verifier as

Parameter	Type	Description
		DD/MM/YYYY. If set to **MMDDYYYY**, the date will be displayed in Verifier as MM/DD/YYYY. If set to **YYYYMMDD**, the date will be displayed in Verifier as YYYY-MM-DD.
FutureInvoiceDateAllowed	YES/NO	If set to **NO** and an invoice date has been extracted, which is ahead of the current date, the Invoice Date field will be set to invalid. If set to **YES**, and the invoice date is in the future, no action will be taken.
DateOnlyInCurrentMonth	YES/NO	If set to **YES** and the extracted invoice date is in a different month, the Invoice Date field will be set to invalid.
PastDays	Number	If populated, the system will set the invoice date to invalid if it is more than the specified number of days in the past from the current date. If left blank, no check is carried out.
MMDDCountries	Freetext	Comma-separated list of countries that use MM/DD/YYYY as the date format preference (e.g. the US).
YYMMDDCountries	Freetext	Comma-separated list of countries in which YY-MM-DD is a standard date format (e.g. Sweden). If the vendor country of origin is included in this list, and the invoice date is read as, for example, **12-01-11**, this will be formatted to **11/01/2012** (DDMM) or **01/11/2012** (MMDD) if the invoice date year is the current year, the previous year, or the following year. If the invoice date year is something else, the format will be assumed to be DD-MM-YY. Caution should be exercised when adding countries to this list.
DeliveryDateRequired	YES/NO	If set to **YES**, the Delivery Date field will be made a mandatory field and the document will be sent to Verifier if no value is extracted. The Verifier user will be permitted to let the field pass blank.
DeliveryDateIncludes ShipDate	YES/NO	If set to **YES**, this will expand the scope of delivery dates appropriate for extraction to include ship dates.
DueDateRequired	YES/NO	If set to **YES**, the Due Date field will be made a mandatory field and the document will be sent to Verifier if no value is extracted. The Verifier user will be permitted to let the field pass blank.

4.1.14 TAX Section

This section contains the configuration settings relating to the extraction of tax, and the automatic determination of tax codes for invoice creation.

Parameter	Type	Description
PrimaryRates	Freetext	Comma-separated list of tax rates expected for invoices processed by the AP Packaged Project. This list is optional, but does assist the project in finding the correct tax value on the document. Values matching the primary rate are considered ahead of values entered as secondary rates.
SecondaryRates	Freetext	Comma-separated list of tax rates expected for invoices processed by the AP Packaged Project. This list is optional, but does assist the project in finding the correct tax value on the document.
ActivateVATCompliance Check	YES/NO	If set to **YES**, the system will activated the VAT registration number compliance check. This means that if value added tax is being charged on the invoice, the vendor and bill-to company VAT registration numbers become mandatory fields. If VAT is being charged in a currency which differs to the local currency of the company code for which the invoice is addressed, then the exchange rate

Parameter	Type	Description
		and/or local VAT amount fields are also mandatory.
VATCheckCompanyCode Exceptions	Freetext	Comma-separated list of company codes that are an exception from the VAT registration number compliance rule. If the **ActivateVATComplianceCheck** parameter is set to **YES**, then any company codes listed in this parameter will not have the check carried out. If **ActivateVATComplianceCheck** is set to **NO**, then only company codes specified in this parameter will have the check carried out.
VendorVATCheckOnly	YES/NO	If this value is set to **YES** and the VAT compliance check has been activated, the system will require the VAT registration number of the vendor, rather than that of both the vendor and the bill-to party.
VendorVATCheckCompany CodeExceptions	Freetext	Comma-separated list of company codes that are an exception to the vendor-only VAT registration number check. If the **VendorVATCheckOnly** parameter is set to **YES**, then, for the company codes listed against this parameter, both the vendor and bill-to VAT registration numbers will be mandatory. If **VendorVATCheckOnly** is set to **NO**, then only the vendor VAT registration number will be required for invoices belonging to a company code specified in this list.
CheckVATCrossBorder	YES/NO	If this value is set to **YES**, and the VAT compliance check has been activated, the system will require entry of both the vendor and the bill-to VAT registration numbers if the tax amount on the invoice is zero, both vendor and company code are based in different EU countries, and the vendor has a VAT registration number set in the vendor extract. The cross-border VAT registration number check operates independently of the vendor-only VAT check.
CrossBorderCompanyCode Exceptions	Freetext	Comma-separated list of company codes that are an exception to the EU cross border VAT registration number check. If the **CheckVATCrossBorder** parameter is set to **YES**, then, for the company codes listed against this parameter, the cross-border VAT registration number check will not be carried out. If the **VendorVATCheckOnly** parameter is set to **NO**, then the cross-border VAT registration number check will only be carried out for invoices belonging to company code specified in this list.
ActivateTaxDetermination	YES/NO	If set to **YES**, the system will activate the tax determination and validation feature. Otherwise, no tax code allocations or validations of those allocations will be carried out.
AlwaysUsePOTaxCode	YES/NO	If set to **YES**, the system will always use the purchase order tax code.
AlwaysUseCalculateTaxFlag	YES/NO	If set to **YES**, the system will always require the ERP system to calculate the tax amount rather than passing the tax amount read from the invoice.
TaxFlagExceptionCompany Codes	Freetext	Comma-separated list of company codes that are an exception to the calculate tax flag rule. If the **AlwaysUseCalculateTaxFlag** parameter is set to **YES**, then company codes listed in this parameter will not use the calculate tax flag. If the **AlwaysUseCalculateTaxFlag** parameter is set to **NO**, then the calculate tax flag will be used for company codes specified in this list.
DeriveTaxCodeForNoPO	YES/NO	Flag to determine whether the system should determine a tax code for a parked non-PO invoice.
ValidateFromDB	YES/NO	Flag to denote whether tax codes should be derived and/or validated against a database table. This table should not be used for countries where tax jurisdictions apply, for example, the US, Brazil and Canada.
SQLConnectionGroup	NN	SQL connection group specifying the tax code database connection string as

Parameter	Type	Description
		set in the *SQL section*. If no connection group is specified, the system will use group **01**.
DBTableName	Freetext	The name of the tax code database table.
CheckForICMSTax	YES/NO	ICMS is a form of sales tax used on Brazilian nota fiscal documents. If this parameter is set to **YES**, then the system will attempt to identify this tax amount on incoming documents and will apply the validations to this field. This parameter should be set to **NO** if the solution is not being used to process Brazilian nota fiscal documents.
DeriveShipToFromCompanyCode	YES/NO	If set to **YES**, the ship-to country will be set to the country in which the company code is based.
ReadPlantFromDB	YES/NO	If set to **YES**, the system will read plant information from a database in order to determine the ship-to location for the goods for tax calculation purposes.
PlantSQLConnectionGroup	NN	SQL connection group specifying the plant database connection string as set in the *SQL section*. If no connection group is specified, the system will use group **01**.
DBPlantTable	Freetext	Name of the database table containing the plant information.
DBPlant	Freetext	Name of the column in the plant database table holding the plant ID.
DBPlantCountry	Freetext	Name of the column in the plant database table holding the country in which the plant is located.
DBPlantState	Freetext	Name of the column in the plant database table holding the state in which the plant is located.
DBPlantTaxJurCode	Freetext	Name of the column in the plant database table holding the tax jurisdiction code associated with the plant's location.
BreakOutHSTForCanada	YES/NO	Flag to indicate whether HST should be captured within the **HST** field as opposed to in the **AmountTax** field.
NN_TaxCode	Freetext	Tax code for the tax code settings group *NN* where *NN* is an incremental number (e.g. 01, 02, 03). This settings group is used for countries where tax jurisdictions exist (e.g. in the US, Brazil and Canada). In such instances, the tax code denotes whether the item is taxable or tax exempt, as well as an instruction for processing (e.g. self-assess). For other countries, the tax table should be used for automatic tax code determination and validation.
NN_Country	Freetext	ISO country code for the country to which the group tax code belongs (e.g. US, CA).
NN_IfTax	Freetext	If there is tax on the invoice and the purchase order line item tax code is set to the value in the **NN_TaxCode** parameter, and the company code country is equal to the value in **NN_Country**, then this value is set as the new tax code for the invoice line.
NN_IfNoTax	Freetext	If there is no tax on the invoice and the purchase order line item tax code is set to the value in the **NN_TaxCode** parameter, and the company code country is equal to the value in **NN_Country**, then this value is set as the new tax code for the invoice line.
NN_VendorState	Freetext	Comma-separated list of vendor states that would trigger the selection of a state-dependent tax code.
NN_ShipToState	Freetext	Comma-separated list of ship-to states that would trigger the selection of a state-dependent tax code.
NN_IfTaxState	Freetext	Tax code for use if the vendor is from a state listed in the **NN_VendorState**

Parameter	Type	Description
		parameter, or the goods were shipped to a state listed in the **NN_ShipToState** parameter and tax is being charged on the invoice. I f both **NN_VendorState** and **NN_ShipToState** are populated, then the vendor state must be in the vendor state list, and the ship-to state must be in the ship-to state list, otherwise the **IfTax** code will be used.
NN_IfNoTaxState	Freetext	Tax code for use if the vendor is from a state listed in the **NN_VendorState** parameter, or the goods were shipped to a state listed in the **NN_ShipToState** parameter and no tax is being charged on the invoice. If both **NN_VendorState** and **NN_ShipToState** are populated, then the vendor state must be in the vendor state list, and the ship-to state must be in the ship-to state list, otherwise the **IfNoTax** code will be used.
NN_PayTaxAsBilled	YES/NO	If there is a tax amount read from the invoice and any of the invoice line tax codes are set to the value held in the **NN_TaxCode** parameter, and this flag is set to **YES**, the system will book the tax amount as billed on the invoice, rather than allow the system to calculate it automatically.
NN_ShortPayIfTax	YES/NO	If there is a tax amount on the invoice and all invoice line item tax codes belong to a group that have this flag set to **YES**, then the invoice will be booked minus the tax.
NN_AccountAssignment Categories	Freetext	Comma-separated list of purchase order line account assignment categories. If no tax code is present on the purchase order line and the company code country is equal to the value held in the **NN_Country** parameter, and the purchase order line has an account assignment present in this list, then the tax code held in **NN_TaxCode** will be defaulted as the initial purchase order line item tax code.

4.1.15 CUR Section

This section contains the configuration settings associated with the invoice currency.

Parameter	Type	Description
DollarSignIsUSD	YES/NO	If no currency has been unambiguously determined from the invoice, but a dollar sign has been found in the OCR text, the currency will be set to **USD** if this flag is set to **YES**.
NN_ISOCode	Freetext	Currency ISO code for the currency settings group *NN* where *NN* is an incremental number (e.g. 01, 02, 03, etc.)
NN_Alias	Freetext	Comma-separated list of aliases for the currency represented by the **NN_ISOCode** parameter. For example, *pounds sterling* is an alias for **GBP**, *us dollars* or *us funds* are aliases for **USD**, *swiss francs* is an alias for **CHF**, etc. If any items on this list are found on the document, then the extracted invoice currency is set to the currency in specified in the **NN_ISOCode** parameter.
NN_AmountPrefix	Freetext	Comma-separated list of values that may precede or follow an amount on the invoice, that represent the currency specified in the **NN_ISOCode** parameter. For example, **US$** would be the prefix for **US$1000.00**.
NN_Symbol	Freetext	Currency symbol associated with the currency held in the **NN_ISOCode** parameter, for example, **$**, **£**, **€**, **¥**, etc. This should only be set to a special character, not a single letter.
NN_Country	Freetext	Country associated with the currency held in the **NN_ISOCode** parameter. If the symbol for the currency settings group is found on the document and

Parameter	Type	Description
		that symbol is unique across all currency settings groups, and the vendor country matches the country held in this setting, then the value held in **NN_ISOCode** is extracted as the invoice currency.
NN_ToleranceGroup	Freetext	Tolerance group for the currency, as defined in the *TOL section* of the project configuration. If no tolerance group is set, then tolerance group **01** is used by default. If tolerance group **01** does not exist, then a tolerance of **0.0001** is used at header level, at line item level and for the validation of tax.
OverrideCurrency	Freetext	If populated, the value set against this parameter will always be set as the invoice currency. This value will not be validated on server side.
DefaultCurrency	Freetext	If populated, this will be set as the invoice currency if no other value is extracted from the document.
DefaultPOCurrency	YES/NO	If no currency can be identified from the invoice, the system will default the currency from the purchase order if this setting is set to **YES**.
DefaultVendorCurrency	YES/NO	If no currency can be identified from the invoice, the system will default the currency for the vendor's country of origin if this setting is set to **YES**.
AllowBlank	YES/NO	If this setting is set to **YES** then a blank currency will be permitted to pass through the AP Packaged Project.
ValidateFromDB	YES/NO	If this setting is set to **YES** then the contents of the *Currency* field in Verifier will be validated against a database to check that the currency is valid.
SQLConnectionGroup	NN	SQL connection group specifying the currency database connection string as set in the *SQL section*. If no connection group is specified, the system will use group **01**.
DBTableName	Freetext	Name of the database table against which the currency is validated. This setting is mandatory if a database lookup is to be performed.
DBColumnName	Freetext	Name of the column in the database table containing the valid currency code. This setting is mandatory if a database lookup is to be performed.
ActivateExtendedValidation	YES/NO	Setting this flag to **YES** will perform an extended validation on a currency determined automatically by the system. The extended validation will compare the system-extracted currency to the following items in the order specified: ▪ Permitted global currencies (see the **GlobalCurrencies** parameter below) ▪ The vendor currency if present in the vendor extract ▪ The currency associated with the vendor country of origin ▪ The currency of the company code ▪ The currency associated with the company code country If the invoice currency does not match any of the above, then the currency will be set to invalid for a user to check within the Verifier application. Errors caused by missing, incomplete or incorrect configuration in connection to company code and country lookups will also set the currency field to invalid. User input in Verifier is excluded from the checks above, and will be assumed to be correct as far as a valid currency is entered. The extended validation is only carried out for invoices classified to the *Generic* subclass.
GlobalCurrencies	Freetext	Comma-separated list of permitted global currencies, used in the extended currency validation as described above. Recommendations for content of this field would be **USD** and **EUR**.

4.1.16 CCO Section

This section contains configuration settings that control the validation of the company code.

Parameter	Type	Description
AllowBlank	YES/NO	Specifies whether the company code is mandatory. If this parameter is set to **NO** a company code is required.
DefaultCompanyCode	Freetext	If populated, this will be set as the company code if no other value is extracted from the document.
ValidateFromDB	YES/NO	If set to **YES**, the company code will be validated against a database.
SQLConnectionGroup	NN	SQL connection group specifying the company code database connection string as set in the *SQL section*. If no connection group is specified, the system will use group **01**.
DBTableName	Freetext	Name of the company code database table. This setting is mandatory if a database lookup is to be performed.
DBColumnName	Freetext	Name of the column in the company code database table that holds the valid company codes. This setting is mandatory if a database lookup is to be performed.
DBCountry	Freetext	Name of the column in the company code database table that holds the country in which the company is legally based.
DBCurrency	Freetext	Name of the column in the company code database table that holds the currency for the company code. This is not mandatory if the currency is going to be validated against a database.
DBVATRegNos	Freetext	Name of the column in the company code database table that holds the VAT registration numbers for which the company is registered. If the company is registered for VAT in more than one country, the database field should contain a comma-separated list.

4.1.17 SRC Section

This section holds the mapping between columns in the vendor extract and the values used internally within the AP Packaged Project.

Parameter	Type	Description
ID	Freetext	Vendor ASSA column name denoting the unique identifier. For ERP systems where a vendor at a unique address is represented by a combination of the vendor ID and the site ID, the formula for the ID column must be set to the result of the following calculation: (*Vendor ID* * 1,000,000) + *Site ID* If the vendor ID or site ID is alphanumeric, the formula should be: *VendorID~SiteID* The delimiter (~ in the above example) is configurable via the **AlphNumSiteSeparator** parameter in the *VND section* of the project configuration. The system will raise a configuration error if no delimiter is specified, is more than one character, does not occur, occurs more than once, or occurs as the first character in the unique ASSA ID column. The site ID must be mapped to the **SRC_VL_SITEID** parameter, and the vendor ID stem must be mapped to the **SRC_VL_EXTERNALVENDORID** parameter. If the ERP system uses an external vendor ID (e.g. the supplier number in Oracle Financials), then this value should be mapped to the **SRC_VL_EXTERNALVENDORID** parameter, and the internal vendor ID stem can remain unmapped, but the vendor ID stem component of the ID field should be the internal ERP system vendor ID.

Parameter	Type	Description
SiteID	Freetext	Vendor ASSA column name denoting the vendor site ID. This should only be mapped if the site ID forms part of the **ID** column above.
Name	Freetext	Vendor ASSA column name denoting the vendor name.
Address1	Freetext	Vendor ASSA column name denoting the first line of the vendor's address.
Address2	Freetext	Vendor ASSA column name denoting the second line of the vendor's address.
City	Freetext	Vendor ASSA column name denoting the vendor's city of origin.
Zip	Freetext	Vendor ASSA column name denoting the vendor zip or postal code.
State	Freetext	Vendor ASSA column name denoting the vendor's state, county or region.
Country	Freetext	Vendor ASSA column name denoting the vendor's country of origin.
POBox	Freetext	Vendor ASSA column name denoting the vendor's PO Box.
POBoxZip	Freetext	Vendor ASSA column name denoting the postal or zip code that relates to the vendor's PO box.
EUMember	Freetext	Vendor ASSA column name denoting whether the vendor's country of origin is a member of the European Union.
Currency	Freetext	Vendor ASSA column name denoting the currency for the vendor's country of origin.
TaxID1	Freetext	Vendor ASSA column name denoting the vendor's primary tax ID.
TaxID2	Freetext	Vendor ASSA column name denoting the vendor's secondary tax ID.
VatRegNo	Freetext	Vendor ASSA column name denoting the vendor VAT registration number. Multiple VAT registration numbers must be presented as a comma-separated list if the vendor is VAT registered in more than one country. Ideally, VAT registration numbers should contain the two-character country prefix, but this is not mandatory.
TaxJurCode	Freetext	Vendor ASSA column name denoting the vendor's tax jurisdiction code for countries where tax jurisdictions are used.
TelNo	Freetext	Vendor ASSA column name denoting the vendor's telephone number.
InvoiceType	Freetext	Vendor ASSA column name denoting whether the vendor is permitted to submit a non-PO invoice.
PaymentMethods	Freetext	Vendor ASSA column name denoting the vendor's payment methods.
BankDetails	Freetext	Vendor ASSA column name denoting the vendor's bank account details. This should be a colon-separated list in the format: *BankAccount,SortCode,ERPBankAccountCode* A sort code is the US equivalent of a routing number.
WithholdingTaxDetails	Freetext	Vendor ASSA column name denoting the vendor's withholding tax details. This should be a colon-separated list in the format: *CompanyCode,WithholdingTaxType,WithholdingTaxCode* If there is withholding tax on the invoice, at time of posting, the system will post the full withholding tax amount to the entry with the relevant company code.
CompanyCodes	Freetext	Vendor ASSA column name denoting a comma-separated list of company codes for which the vendor is valid.
UtilityFlag	Freetext	Vendor ASSA column name denoting whether the vendor is a utility vendor

Parameter	Type	Description
		or not.
PORSubscriberNo	Freetext	Vendor ASSA column name denoting the vendor's Post Office Reference number as used in Switzerland.
ExternalVendorID	Freetext	Vendor ASSA column name denoting the vendor's external ID (e.g. the supplier number as used in Oracle Financials). If no external vendor ID is used by the ERP system, but the combination of a vendor ID and a site ID is used to identify a unique vendor address, this column must be mapped to the vendor ID stem.
SiretID	Freetext	Vendor ASSA column name that represents the vendor's SIRET ID number, as assigned to French vendors.
EUMemberAlias	Freetext	Vendor ASSA column name that contains the value that denotes a positive identification of the vendor as a European Union member state.

4.1.18 VND Section

This section contains settings for validating an extracted vendor number.

Parameter	Type	Description
ValidateFromASSA	YES/NO	Denotes whether an extracted vendor ID should be validated against the Associative Search Engine pool. Oracle recommends that these settings should always be set to **YES**.
CheckConditionVendors	YES/NO	If set to **YES**, an alternate vendor ID will be allowed for a PO-based invoice if it matches a vendor held within a condition record (e.g. for planned freight) behind a purchase order line.
AlphNumSiteSeparator	Freetext	Special character used to separate a vendor ID and site ID in the unique ID column in the vendor ASSA pool.
IgnorePOVendor	YES/NO	If set to **YES**, the system will always use the vendor determined by the system rather than defaulting to the remit-to vendor set on the purchase order. Moreover, it will no longer be a requirement that the purchase order vendor has to be found on the document for the purchase order to be set to valid. Instead, both values are obtained and validated independent of one another, although an otherwise invalid vendor will be set to valid if it is corroborated by the vendor on the purchase order. Hence, the invalid reasons for **PO VENDOR IS NOT INVOICE VENDOR** and **THIRD PARTY FREIGHT** no longer apply.
UseASSAIfPOVendorInvalid	YES/NO	If set to **YES**, then the system will evaluate the vendor determined by the system if none of the vendors on the purchase order can be validated against the document. If the system-determined vendor can be validated, this will be displayed in the vendor ID field, and the invalid reason will automatically be set to **PO VENDOR IS NOT INVOICE VENDOR**. If the system-determined vendor cannot be validated, it will be displayed in the field, but will be set to invalid. The content of the invalid reason field will not be changed. This parameter will only take effect if the **IgnorePOVendor** parameter is set to **NO**.
DefaultCountry	Freetext	If no country column is available in the vendor extract used by the Vendor ASSA field, or the value in the country column is blank, a default country for all vendors may be specified here. This should be a two-character ISO country code.
UseBillToBasedVendor	YES/NO	If this value is set to **YES** and a bill-to name field has been captured from the

Parameter	Type	Description
Extraction		invoice, the system will dynamically modify the search area for the invoice vendor to include the area of the first page of the document above where the bill-to name was detected, along with the bottom ten percent of the first page of the document.
RefineVendorExtraction	YES/NO	If this value is set to **YES**, the system will manipulate the weighting of the candidates for the vendor, adding additional confidence if the vendor name, the vendor street address, the vendor zip code, the vendor VAT registration number and the vendor SIRET ID can be found physically on the document. The ASSA-delivered weightings will be increased as follows: ▪ Zip only +10% ▪ Zip + Vendor Name +20% ▪ Zip + VAT Code +20% ▪ Zip + Street Name +20% ▪ Zip + Street Name + Vendor Name +30% ▪ Zip + Street Name + VAT Reg +30% ▪ Zip + Street Name + VAT Reg +30% ▪ Zip + Street Name + VAT Reg + Vendor Name +40% An additional 20% is added to any of the above if the SIRET ID is found on the document (for French vendors), or a unique vendor identifier.
CheckNameForNOPO	YES/NO	If this parameter is set to **YES**, the system will not allow a vendor ID to validate automatically unless either the vendor name or the vendor VAT registration number is found in the OCR text of the document. This applies to all documents except PO-based invoices where a valid PO number is found and the **IgnorePOVendor** parameter is set to **NO**.

4.1.19 TAB Section

This section contains configuration settings that control the validation of the invoice line item data (table extraction validation).

Parameter	Type	Description
ExtractLineItems	YES/NO	If set to **YES**, line items will be extracted.
SkipForService	YES/NO	If set to **YES**, line items will not be mandatory for invoices relating to a service purchase order.
LineTotalOnlyForService	YES/NO	If set to **YES**, only the line item total value is required for the invoices relating to a service purchase order.
SkipForNoPO	YES/NO	If set to **YES**, line items will not be mandatory for no-PO invoices.
SkipForCredit	YES/NO	If set to **YES**, line items will not be mandatory for credit memos.
SkipForMIRA	YES/NO	If set to **YES**, line items will not be mandatory for invoices which are a 1:1 match with the total purchase order value or the value of all goods receipts not yet invoiced for the entire purchase order.
SkipForUnreleasedPO	YES/NO	If set to **YES**, line items will not be mandatory for invoices relating to a purchase order that has not yet been released.
SkipForInvalidPO	YES/NO	If set to **YES**, the system will not require line items if the **MISSING/ INVALID PURCHASE ORDER** invalid reason has been selected by the user.
SkipForInvalidVendor	YES/NO	If set to **YES**, the system will not require line items if the **VENDOR NOT FOUND** invalid reason has been selected by the user.
SkipForUtilityVendor	YES/NO	If set to **YES**, the system will not require line items if the vendor is a utility vendor. This will save considerable processing time if utility invoices are large

Parameter	Type	Description
		documents with multiple backing sheets that the system does not need to evaluate as potential line items that require extraction.

4.1.20 MSC Section

This section controls the identification and handling of miscellaneous charges (e.g. freight, customs charges, fuel surcharges etc.) that may appear on an invoice document, both at header and at line item level

Parameter	Type	Description
UnplannedThreshold	Numeric	Amount in the invoice currency which represents the maximum value the sum of all unplanned miscellaneous charges may reach before the system takes a course of action
BlockIfOverThreshold	YES/NO	If set to **YES**, if the total sum of all unplanned miscellaneous charges on the invoice exceeds the set threshold, the system will set a block on a created invoice document.
BlockCode	Freetext	Block code to be applied to an invoice for reasons of excessive unplanned miscellaneous charges
ThirdPartyFreightCode	Freetext	Identifies the miscellaneous charge category from which the rules for posting third party freight invoices should be lifted.
HandleMiscChargesFor Services	YES/NO	Indicates whether the miscellaneous charge processing logic should also be applied to invoices that relate to service purchase orders
ValidateFromDB	YES/NO	If set to **YES**, the system will look into a database table in order to determine the correct general ledger entry for a miscellaneous charge.
SQLConnectionGroup	Freetext	SQL connection group specifying the miscellaneous charge database connection string as set in the *SQL section*. If no connection group is specified, the system will use group **01**.
DBTableName	Freetext	Name of the database table containing the GL coding strings to be used for miscellaneous charges. The **WFR_AP_Tables_Create_<***database***>.sql** script provided with the AP Packaged Project can be used to create (but not populate) an **OFRMISCACC** table to be used here.
NN_Type	Freetext	Miscellaneous charge type for charge group *NN*.
NN_Code	Freetext	Internal code for charge group *NN*.
NN_HeaderField	Freetext	Name of the header field that contains miscellaneous charges belonging to the charge group.
NN_Alias	Freetext	Comma-separated list of identifying strings so that the system can recognize miscellaneous charges specified at the line item level via the article description.
NN_LineType	Freetext	ERP system line type for the charge group *NN*.
NN_AlwaysBookTo Unplanned	YES/NO	If set to **YES**, the system will handle all charges found on this document that are identified as belonging to charge group *NN* as unplanned miscellaneous charges.
NN_AlwaysBookTo Planned	YES/NO	If set to **YES**, the system will look to book all miscellaneous charges identified as belonging to charge group *NN* to a planned charge on the purchase order.
NN_AlwaysBookToGL Account	YES/NO	If set to **YES**, the system will create a general ledger account entry for all miscellaneous charges identified as belonging to charge group *NN*.
NN_BookToUnplanned IfNoPlanned	YES/NO	If **NN_AlwaysBookToPlanned** is set to **YES** and no conditions or line types could be found on the purchase order, but charges belonging to this charge

Parameter	Type	Description
		group were found on the invoice, the system will handle these charges as unplanned if this setting is set to **YES**.
NN_BookToGLAccount IfNoPlanned	YES/NO	If **NN_AlwaysBookToPlanned** is set to **YES** and no conditions or line types could be found on the purchase order, but charges belonging to this charge group were found on the invoice, the system will create a general ledger entry for those charges.
NN_GLAccount	Freetext	Default general ledger account code to be used for miscellaneous charge general ledger entries.
NN_GetCostObjectFrom POLine	YES/NO	Flag to denote whether the cost object information to complete the general ledger entry for the miscellaneous charge should be read from a paired purchase order line item. The cost object can either be a cost center, an internal order or a project, and the system will read this from the first paired invoice line which has one of these cost object assignments.
NN_DefaultCostCenter	Freetext	Default cost center to be used for miscellaneous charge general ledger entries.
NN_ DefaultProfitCenter	Freetext	Default profit center to be used for miscellaneous charge general ledger entries.
NN_DefaultTaxCode	Freetext	Default tax code to be used for miscellaneous charge general ledger entries. For countries and ERP systems that use tax jurisdictions, the corresponding tax jurisdiction code is read from the first paired invoice line, which denotes the ship-to address for the goods.

4.1.21 UOM Section

This section contains mapping information between a unit of measure that might appear on an invoice to a unit of measure ISO-code as used by the downstream ERP system.

Parameter	Type	Description
NN_ISOCode	Freetext	Unit of measure ISO-Code/internal ERP system format for UOM group *NN*.
NN_Alias	Freetext	Comma-separated list of aliases that the unit of measure may appear on the invoice under for UOM group *NN*.
NN_ExportValue	Freetext	This parameter is optional and is only used when the *Export Custom Unit of Measure Value to XML* feature is enabled, and the extracted unit of measure value exists in comma-separated list defined in the *NN_Alias* parameter. It defines the custom value that should be written to the <*UOM*> tag of the XML output for UOM group *NN*.
SQLConnectionGroup	NN	SQL connection group specifying the unit of measure conversion table database connection string as set in the *SQL section*. If no connection group is specified, the system will use group **01**.
DBTableName	Freetext	Name of the unit of measure conversion database table. This setting is mandatory if a database lookup is to be performed for unit of measure conversion.
DBMaterialNo	Freetext	Technical name of the column in the unit of measure conversion table that represents the item material number.
DBExternalUOM	Freetext	Technical name of the column in the unit of measure conversion table that represents the external unit of measure read from the invoice from which conversion is to take place.
DBNumerator	Freetext	Technical name of the column in the unit of measure conversion table that represents the numerator component of the unit of measure conversion ratio.
DBDenominator	Freetext	Technical name of the column in the unit of measure conversion table that represents the denominator component of the unit of measure conversion

Parameter	Type	Description
		ratio.
DBBaseUOM	Freetext	Technical name of the column in the unit of measure conversion table that represents the material base unit of measure (i.e. the destination unit of measure).

4.1.22 TOL Section

In this section, the tolerances used to validate the extracted amounts can be specified. Tolerances are set in tolerance groups, which, in turn, can be assigned to specific currencies in the *CUR section* of the project configuration.

Each group has a group code and a set of three tolerances controlling the permitted deviations at header level, at line item level and for the purposes of validating tax amounts.

Parameter	Type	Description
NN_HeaderTolerance	Number	Tolerance value for the invoice header amounts in the invoice currency.
NN_TableRowTolerance	Number	Tolerance value for each individual line item in the invoice currency.
NN_TaxTolerance	Number	Tolerance value for comparing the invoice tax amount with the system calculated tax amount based on the line item tax codes.
NN_NoDecimalPlaces	YES/NO	Indicates whether currencies assigned to this tolerance group are relevant for decimal places. This is appropriate for currencies where a single unit of that currency is the smallest currency unit in current use (i.e. there are no active sub-units, such as cent, penny, etc.). For example the Japanese Yen or the Hungarian Forint. If **10.400** is extracted and the currency is Hungarian Forints (HUF), and **HUF** is assigned to a tolerance group with this parameter set to **YES**, then the value will be formatted to **10400**. If the parameter is set to **NO**, it will be formatted to **10.40**. However, if **10.40** is extracted (i.e. 2 decimal places), then the formatting will stay at **10.40**.

4.1.23 IVR Section

This section contains the configuration settings for the invalid reason field in the AP Packaged Project, which controls the options that are available and how the system responds to these options.

Parameter	Type	Description
DefaultText	Freetext	Default invalid reason (e.g. **NONE**).
DefaultExportCode	Freetext	Export code associated with the default invalid reason (e.g. **0**).
NN_Rule	Freetext	Rule ID for the invalid reason belonging to group *NN*. The rule governs how Verifier behaves as a result of a particular invalid reason being selected. Refer to *Section 3.1.28: Invalid Reason* for a description of the available rules and their effect on the document.
NN_VerifierDisplay	Freetext	Invalid reason message displayed in Verifier for group *NN*.
NN_ExportCode	Freetext	Invalid reason code exported by the AP Packaged Project if the invalid reason field is set belonging to group *NN*.

4.1.24 ERR Section

This section contains the field error messages displayed to the user through the Verifier application. Each setting takes the format, where *NNN* is the error code number:

```
ERR_VL_NNN=<Error Text>
```

The error code number cannot be changed, but the error text can be changed to whatever is appropriate for the client, for example, the language of the error message.

Parameter	Type	Description
001 – 199	Freetext	Error message text that can be customized as needed, such as providing error messages in a language other than English.

4.1.25 INF Section

This section contains the text for the information boxes displayed in the Verifier application.

Parameter	Type	Description
DialogHeader	Freetext	Information box title bar text.
NNN	Freetext	Information message for code *NNN*. Within the information message, text symbols **[VEN]** and **[CUR]** are used to represent the current vendor ID and currency respectively.
DisableMIRAPopup	YES/NO	If set to **YES**, this will disable the pop-up box in Verifier informing the user that line items are not required as there is a 1:1 match on value between the invoice and the purchase order.
DisableCurrencyPopup	YES/NO	If set to **YES**, the system will not prompt the Verifier user if the currency is about to be over-written with a vendor or purchase order default currency if the vendor or PO is changed.

4.1.26 EXP Section

This section holds the configuration settings relating to the AP Project data export options.

Parameter	Type	Description
RedoAllExports	YES/NO	If set to **YES**, the system will carry out all export options that have been activated, even if that export has been carried out before. For example, if 4 export options are activated, 3 are completed and the last one fails, then the document will go to status 750 denoting an export failure. If the flag is set to **NO**, upon retrying, only the failed export will be carried out. If the flag is set to **YES**, then all 4 exports will be performed again.
OutputStandardResultsFile	YES/NO	If set to **YES**, the system will output a standard results file into the export directory. The file extension is set to **.txt**.
Filename	Freetext	This setting controls the name of the standard results file. This can be set either to **URN**, which will name the file according to the component of the image filename mapped in the *IMP section*, or left blank. If left blank, or set to anything else, the filename will be set to the same name as the document filename. The file extension is **.txt**.
DefaultExportPath	Freetext	UNC path to the export directory that is used as the default if no export directory is specified in the RTS instance properties.
OutputTiffFile	YES/NO	If set to **YES**, the system will output a TIFF file of the document image in the export directory.
TiffName	Freetext	This setting controls the name of the output TIFF file. This can be set either to **URN**, which will name the file according to the component of the image

Parameter	Type	Description
		filename mapped in the *IMP section*, or left blank. If left blank, or set to anything else, the filename will be set to the same name as the document filename.
TiffDPI	Number	Specifies the DPI of the outputted TIFF image. In case of a blank or invalid entry, the default TIFF resolution is 300 dpi.
TiffFormat	Freetext	Compression format of the outputted TIFF file. The valid options for this parameter are: ▪ G4FAX Group 4 compression ▪ G3FAX Group 3 compression ▪ LZWFAX LZW Compression ▪ HUFFAX HUF Compression The default compression in the event of a blank or invalid entry is **G4FAX**.
RedactInvoiceNumber	YES/NO	If set to **YES**, the system will redact the invoice number on an outputted TIFF image.
OutputPDF	YES/NO	If set to YES, the system will output a searchable PDF file for each document.
PDFName	Freetext	This setting controls the name of the output PDF file. This can be set either to **URN**, which will name the file according to the component of the image filename mapped in the *IMP section*, or left blank. If left blank, or set to anything else, the filename will be set to the same name as the document filename.
OutputDateFormat	Freetext	Output date format for export. The valid options for this parameter are: ▪ DDMMYYYY ▪ MMDDYYYY ▪ YYYYMMDD This setting applies to database output and all flat file exports.
OutputDateSeparator	Freetext	Separator to be used in the date format, for example, a hyphen, period/full-stop or forward slash. This parameter can be left blank for no separator. This setting applies to database output and all flat file exports.
CustomExport	YES/NO	Flag to indicate whether a custom export should be carried out, as specified in script the user exit **UserExitCustomExport**.
ExportToDB	YES/NO	Flag to denote whether the extracted data should be written to a database upon document export.
SQLConnectionGroup	NN	SQL connection group specifying the export database connection string as set in the *SQL section*. If no connection group is specified, the system will use group **01**. The header and line item tables must exist in the same database.
DBKey	Freetext	Key to be used for each new database record inserted into the header table. This value can be set to **URN**, in which case the portion of the filename mapped to the URN in the *IMP section* is used. Any other value or a blank entry will use the entire document filename. Before inserting a new record, the system will delete any existing records with the same key.
DBHeaderTable	Freetext	Name of the table in the database where the header fields should be inserted.
DBHeaderKey	Freetext	Name of the column in the header export database table which represents the key field for the header record.
DBHeaderOperation	Freetext	Operation to be performed on the database record. If set to **INSERT**, the system will insert a new record. If set to **UPDATE**, the system will update an

Parameter	Type	Description
		existing record with the same database key. There is no default setting for this parameter.
DBHCDocumentType	Freetext	Name of the header export database column that holds the document type.
DBHCInvoiceType	Freetext	Name of the header export database column that holds the invoice type.
DBHCPOType	Freetext	Name of the header export database column that holds the PO type.
DBHCERPDocType	Freetext	Name of the header export database column that holds the ERP system document type.
DBHCCompanyCode	Freetext	Name of the header export database column that holds the company code.
DBHCInvoiceNumber	Freetext	Name of the header export database column that holds the invoice number.
DBHCInvoiceDate	Freetext	Name of the header export database column that holds the invoice date. This field is outputted in accordance with the export **OutputDateFormat** and **OutputDateSeparator** settings.
DBHCVendorID	Freetext	Name of the header export database column that holds the vendor ID.
DBHCInternalVendorID	Freetext	Name of the header export database column that holds the internal vendor ID.
DBHCSiteID	Freetext	Name of the header export database column that holds the vendor site ID.
DBHCVendorAccountGroup	Freetext	Name of the header export database column that holds the ERP vendor account group.
DBHCVendorName	Freetext	Name of the header export database column that holds the vendor name.
DBHCBillToName	Freetext	Name of the header export database column that holds the bill-to name.
DBHCPO	Freetext	Name of the header export database column that holds the purchase order number.
DBHCTax	Freetext	Name of the header export database column that holds the invoice tax amount.
DBHCWithholdingTax	Freetext	Name of the header export database column that holds the invoice withholding tax amount.
DBHCMisc	Freetext	Name of the header export database column that holds the invoice miscellaneous charge amount at header level.
DBHCDiscount	Freetext	Name of the header export database column that holds the invoice header level discount.
DBHCInvoiceTotal	Freetext	Name of the header export database column that holds the invoice total amount.
DBHCPST	Freetext	Name of the header export database column that holds the PST/QST tax amount.
DBHCHST	Freetext	Name of the header export database column that holds the HST tax amount.
DBHCICMS	Freetext	Name of the header export database column that holds the Brazilian ICMS tax amount
DBHCUnplannedFreight	Freetext	Name of the header export database column that holds the unplanned miscellaneous charge amount from the invoice. This will default to the freight amount specified on the invoice if line pairing is not carried out, or no configuration exists for miscellaneous charges.
DBHCCurrency	Freetext	Name of the header export database column that holds the invoice currency.
DBHCPORNumber	Freetext	Name of the header export database column that holds the Post Office Reference (POR) number. This is used in Switzerland.

Parameter	Type	Description
DBHCPORSubscriberNo	Freetext	Name of the header export database column that holds the Swiss POR subscriber number for the vendor.
DBHCPaymentReference	Freetext	Name of the header export database column that holds the invoice payment reference (e.g. the Norwegian KID number).
DBHCBankAccount	Freetext	Name of the header export database column that holds the vendor's designated bank account.
DBHCBankAccountCode	Freetext	Name of the header export database column that holds the ERP system code for the vendor's designated bank account.
DBHCExchangeRate	Freetext	Name of the header export database column that holds the invoice currency exchange rate.
DBHCEmployeeID	Freetext	Name of the header export database column that holds the ERP system employee ID.
DBHCEmployeeName	Freetext	Name of the header export database column that holds the name of the employee.
DBHCAccountNumber	Freetext	Name of the header export database column that holds the account number for which the invoice relates (e.g. as found on utility bills).
DBHCScanDate	Freetext	Name of the header export database column that holds the invoice scan date. This field is outputted in accordance with the export **OutputDateFormat** and **OutputDateSeparator** settings.
DBHCBatchName	Freetext	Name of the header export database column that holds the name of the batch into which the invoice was scanned.
DBHCPriorityFlag	Freetext	Name of the header export database column that holds the invoice priority flag.
DBHCURN	Freetext	Name of the header export database column that holds the document URN.
DBHCInvalidReason	Freetext	Name of the header export database column that holds the text for the invalid reason as set by the Verifier user.
DBHCInvalidReasonCode	Freetext	Name of the header export database column that holds the export code relating to the invalid reason as set in the *IVR section*.
DBHCDocumentLink	Freetext	Name of the header export database column that holds a link to the document image, whether it resides in a third-party external archive, within the AP Project Reporting database, or in a storage directory.
DBHCPaymentMethod	Freetext	Name of the header export database column that holds the vendor payment method code.
DBHCWithholdingTaxCode	Freetext	Name of the header export database column that holds the vendor withholding tax code.
DBHCAlternatePayee	Freetext	Name of the header export database column that holds the alternate payee associated with the vendor.
DBHCPermittedPayee	Freetext	Name of the header export database column that holds the permitted payee associated with the vendor.
DBHCStatus	Freetext	Name of the header export database column that holds the current document status. This is subsequently overwritten with the new document status, the corresponding value of which is set in **DBStatusExported**.
DBHCERPPOType	Freetext	Name of the header export database column that holds JD Edwards purchase order type code for the purchase order captured at header level.
DBHCBusinessUnit	Freetext	Name of the header export database column that holds the PeopleSoft purchasing business unit for the purchase order number captured at header level.

Parameter	Type	Description
DBHCDeliveryNote	Freetext	Name of the header export database column that holds the delivery note number of the vendor.
DBHCDeliveryDate	Freetext	Name of the header export database column that holds the invoice delivery date.
DBHCDueDate	Freetext	Name of the header export database column that holds the invoice due date.
DBHCISR	Freetext	Name of the header export database column that holds the ISR retention amount.
DBStatusExported	Freetext	Value or code that denotes a document successfully exported from the AP Packaged Project.
DBLineItemsTable	Freetext	Name of the database table into which the invoice line item information should be written. The line item database export always occurs as an insert, and any existing lines for the same document record will be deleted beforehand.
DBLineItemsKey	Freetext	Name of the column in the line item export database table which, along with the header level key, represents the key field for each invoice line item record. This is typically set to the invoice line item number.
DBTablePO	Freetext	Name of the column in the line item export database table that holds the purchase order number to which the invoice line item relates.
DBTablePOLine	Freetext	Name of the column in the line item export database table that holds the purchase order line item number to which the invoice line item relates.
DBTableDescription	Freetext	Name of the column in the line item export database table that holds the invoice line item description.
DBTableMaterialNo	Freetext	Name of the column in the line item export database table that holds the invoice line item material number.
DBTableMaterialGroup	Freetext	Name of the column in the line item export database table that holds the invoice line item material group.
DBTableQuantity	Freetext	Name of the column in the line item export database table that holds the invoice line item quantity.
DBTableUOM	Freetext	Name of the column in the line item export database table that holds the invoice line item order unit of measure.
DBTableUnitPrice	Freetext	Name of the column in the line item export database table that holds the invoice line item unit price.
DBTablePUOM	Freetext	Name of the column in the line item export database table that holds the invoice line item order price unit of measure.
DBTableQtyPUOM	Freetext	Name of the column in the line item export database which holds the invoice line item quantity in the order price unit of measure.
DBTableTotal	Freetext	Name of the column in the line item export database table that holds the invoice line item total amount.
DBTableTaxCode	Freetext	Name of the column in the line item export database table that holds the invoice line item tax code.
DBTableTaxJurCode	Freetext	Name of the column in the line item export database table that holds the invoice line item tax jurisdiction code.
DBTableConditionType	Freetext	Name of the column in the line item export database table that holds the invoice line item condition type.
DBTableConditionStepNo	Freetext	Name of the column in the line item export database table that holds the invoice line item condition step number.

Parameter	Type	Description
DBTableConditionCount	Freetext	Name of the column in the line item export database table that holds the invoice line item condition count.
DBTableFreightVendor	Freetext	Name of the column in the line item export database table that holds the invoice line item freight vendor.
DBTableGRDocNo	Freetext	Name of the column in the line item export database table that holds the number of the goods receipt document to which the invoice line relates.
DBTableGRDocYear	Freetext	Name of the column in the line item export database table that holds the year of the goods receipt document to which the invoice line relates.
DBTableGRDocItem	Freetext	Name of the column in the line item export database table that holds the item number of the goods receipt document to which the invoice line relates.
DBTableSheetNo	Freetext	Name of the column in the line item export database table that holds the service entry sheet to which the invoice line relates.
DBTableSheetItem	Freetext	Name of the column in the line item export database table that holds the item number of the service entry sheet to which the invoice line relates.
DBTableSubDebIndicator	Freetext	Name of the column in the line item export database table that holds the flag denoting a subsequent debit or credit for the invoice line item.
DBTableLineType	Freetext	Name of the column in the line item export database table that holds the invoice line item type or category.
DBTableChargeCode	Freetext	Name of the column in the line item export database table that holds the invoice line item charge code.
DBTableChargeCodeID	Freetext	Name of the column in the line item export database table that holds the invoice line item charge code ID.
DBTableDistillerLine	Freetext	Name of the column in the line item export database table that holds the original line number for the item in the table captured by the AP Packaged Project. This value is only populated if line pairing is enabled and then only for lines where the line pairing was not successful. Internally, the project table lines use a zero-based index. This means that the first line item in the project table is line zero internally. This field is populated with an incremented value so that a value of **1** will be output for project line item 1, a value of **2** will be output for project line item 2, etc.
DBTablePlant	Freetext	Name of the column in the line item export database table that holds the plant to which the invoice line item relates.
DBTableCompanyCode	Freetext	Name of the column in the line item export database table that holds the company code.
DBTableERPPOType	Freetext	Name of the column in the line item export database table that holds the JD Edwards purchase order type.
DBTableBusinessUnit	Freetext	Name of the column in the line item export database table that holds the PeopleSoft purchasing business unit.
DBTableTaxRate	Freetext	Name of the column in the line item export database table that holds the tax rate associated with the line item.
DBGLItemsTable	Freetext	Name of the database table into which general ledger account line entries should be written.
DBGLItemsKey	Freetext	Name of the column in the general ledger export database table which, along with the header level key, represents the key field for each general ledger record. This is typically set to the general ledger item number.

Parameter	Type	Description
DBGLTableGLAccount	Freetext	Name of the column in the general ledger export database table that holds the general ledger account number.
DBGLTableCompanyCode	Freetext	Name of the column in the general ledger export database table that holds the company code for the general ledger entry.
DBGLTableCostCenter	Freetext	Name of the column in the general ledger export database table that holds the cost center for the general ledger entry.
DBGLTableInternalOrder	Freetext	Name of the column in the general ledger export database table that holds the internal order for the general ledger entry.
DBGLTableWBSElem	Freetext	Name of the column in the general ledger export database table that holds the Work Breakdown Structure for the general ledger entry.
DBGLTableNetwork	Freetext	Name of the column in the general ledger export database table that holds the network for the general ledger entry.
DBGLTableActivity	Freetext	Name of the column in the general ledger export database table that holds the network activity for the general ledger entry.
DBGLTableProfitCenter	Freetext	Name of the column in the general ledger export database table that holds the profit center for the general ledger entry.
DBGLTableSalesOrder	Freetext	Name of the column in the general ledger export database table that holds the sales order for the general ledger entry.
DBGLTableSalesOrderItem	Freetext	Name of the column in the general ledger export database table that holds the sales order item for the general ledger entry.
DBGLTableBusinessArea	Freetext	Name of the column in the general ledger export database table that holds the business area for the general ledger entry.
DBGLTableTaxCode	Freetext	Name of the column in the general ledger export database table that holds the tax code for the general ledger entry.
DBGLTableTaxJurCode	Freetext	Name of the column in the general ledger export database table that holds the tax jurisdiction code for the general ledger entry.
DBGLTableTotal	Freetext	Name of the column in the general ledger export database table that holds the total value of the general ledger entry in the invoice currency.
DBGLTableIndicator	Freetext	Name of the column in the general ledger export database table that holds the debit/credit indicator for the general ledger entry.
OutputXMLFile	YES/NO	If set to **YES**, the system will output an XML file to the export directory configured on the RTS export instance. If no directory is configured, then the **DefaultExportPath** parameter is used. If this is not configured either, then the XML export will fail and the batch will be sent to status 750.
XMLFilename	Freetext	This setting controls the name of the XML output file. This can be set either to **URN**, which will name the file according to the component of the image filename mapped in the *IMP section*, or left blank. If left blank, or set to anything else, the filename will be set to the same name as the document filename.
XMLFileType	Freetext	File extension applied to the XML file. The dot should not be specified in the parameter value, for example: `EXP_VL_XMLFileType=XML` If left blank, the file extension will default to **XML**.
XMLEncodingHeader	Freetext	XML file coding header that will form the first line in the XML file. For example, setting the value as **<xml version="1.0" encoding="UTF-16"?>** would produce an XML file that will support non-Western characters such as letter from the Russian, Greek and Chinese alphabets.
XMLStatusExported	Freetext	Value or code that denotes a document successfully exported from the AP

Parameter	Type	Description
		Packaged Project. This value is subsequently placed as the output against the **XMLStatus** field.
XMLFileHeader	Freetext	This denotes the value of the file header tag in the XML file. This value will default to **BrainwareDocument** if left blank.
XMLDocClass	Freetext	This denotes the tag to be used to mark the project document class in the document header section of the XML output file. If set to **DocClass**, the output will be: `<DocClass>project class name</DocClass>` If this parameter is left blank, the project document class will not be written into the XML file.
XMLScanDate	Freetext	This denotes the tag to be used to mark the Scan Date (as mapped in the *IMP section* of the project configuration) in the document header section of the XML output file. If set to **ScanDate**, then the output will be: `<ScanDate>scan date</ScanDate>` If this parameter is left blank, the scan date will not be written into the XML file. The format of the scan date, along with the desired separator is configured using the **OutputDateFormat** and **OutputDateSeparator** parameters.
XMLURN	Freetext	This denotes the tag to be used to mark the document URN (as mapped in the *IMP section* of the project configuration) in the document header section of the XML output file. If set to **URN**, the output will be: `<URN>document urn</URN>` If this parameter is left blank, the document URN will not be written into the XML file.
XMLBatchName	Freetext	This denotes the tag to be used to mark the document batch name (as mapped in the *IMP section* of the project configuration) in the document header section of the XML output file. If set to **BatchName**, the output will be: `<BatchName>document batch name</BatchName>` If this parameter is left blank, the document batch name will not be written into the XML file.
XMLPriorityFlag	Freetext	This denotes the tag to be used to mark the document priority flag (as mapped in the *IMP section* of the project configuration) in the document header section of the XML output file. If set to **PriorityFlag**, the output will be: `<PriorityFlag>priority flag</PriorityFlag>` If this parameter is left blank, the priority flag will not be written into the XML file.
XMLDocumentLink	Freetext	This denotes the tag to be used to mark the document link in the document header section of the XML output file. The document link could be a file path to the image in the batch directory, the file path to the image in a storage directory, or an HTTP link to the image in the Visibility archive. This is dependent on settings in the *REP section* of the project configuration. If set to **DocLink**, the output will be: `< DocLink >document link</ DocLink >` If this parameter is left blank, the document link will not be written into the XML file.
XMLStatus	Freetext	This denotes the tag to be used to mark the document status code (as defined by the value of **XMLStatusExported**) in the document header section of the

Parameter	Type	Description
		XML output file. If set to **Status**, the output will be: `<Status>document status code</Status>` If this parameter is left blank, the document status code will not be written into the XML file.
XMLInvoiceHeader	Freetext	This denotes the value of the tag marking the invoice header section in the XML file. This value will default to **InvHeader** is left blank.
XMLHC*xxx*	Freetext	This denotes the tag to be used to mark the invoice header output field represented by *xxx*. For example, **XMLHC*InvoiceNumber*** is the parameter for the invoice number. If the **XMLHXInvoiceNumber** parameter is set to **invoiceNumber**, and **1234** is extracted as the invoice number, then the following line will be written into the invoice header section of the XML file: `<invoiceNumber>1234</invoiceNumber>` If this parameter is left blank, the corresponding field will not be written into the invoice header section of the XML file. The following fields are available for output: ▪ XMLHCDocumentType ▪ XMLHCInvoiceType ▪ XMLHCPOType ▪ XMLHCERPDocType ▪ XMLHCCompanyCode ▪ XMLHCInvoiceNumber ▪ XMLHCInvoiceDate ▪ XMLHCVendorID ▪ XMLHCInternalVendorID ▪ XMLHCSiteID ▪ XMLHCVendorAccountGroup ▪ XMLHCVendorName ▪ XMLHCBillToName ▪ XMLHCPO ▪ XMLHCTax ▪ XMLHCWithholdingTax ▪ XMLHCMisc ▪ XMLHCDiscount ▪ XMLHCInvoiceTotal ▪ XMLHCPST ▪ XMLHCHST ▪ XMLHCICMS ▪ XMLHCUnplannedFreight ▪ XMLHCCurrency ▪ XMLHCPORNumber ▪ XMLHCPORSubscriberNo ▪ XMLHCPaymentReference ▪ XMLHCBankAccount ▪ XMLHCBankAccountCode ▪ XMLHCExchangeRate ▪ XMLHCEmployeeID ▪ XMLHCEmployeeName ▪ XMLHCAccountNumber ▪ XMLHCInvalidReason ▪ XMLHCInvalidReasonCode ▪ XMLHCPaymentMethod ▪ XMLHCWithholdingTaxCode ▪ XMLHCAlternatePayee ▪ XMLHCPermittedPayee ▪ XMLHCERPPOType ▪ XMLHCBusinessUnit ▪ XMLHCDeliveryNote ▪ XMLHCDeliveryDate ▪ XMLHCDueDate ▪ XMLHCISR

4.1.27 LPR Section

The section holds the settings that are used by the system when carrying out the line pairing operation at the point of document export.

Parameter	Type	Description
DoLinePairing	YES/NO	Flag to indicate whether the line pairing operation should be carried out at the point of document export.
DoLinePairingForService	YES/NO	Flag to indicate whether line pairing should be carried out for invoices that relate to a service purchase order.
CheckForMultiplePOs	YES/NO	Flag to indicate whether the system should consider multiple purchase order numbers on the document at time of line pairing.

Parameter	Type	Description
DescriptionThreshold	Number	This value expressed as a percentage (i.e. **30** = 30%) denotes how confident the system needs to be to pair a line item based on the fuzzy match on description. A setting of **100** requires an exact match between the invoice line and purchase order line descriptions. For example, if the PO line item description were **BROWN HATS**, then a 100% match would be achieved if the invoice line item description were **BROWN HATS**, **brown hats**, **BROWN** or **HAT**. For information purposes, **BROWN HAT** would be a 99.99% match and **HATS BROWN** would be an 85% match. A setting of zero would mean that the closest match to the invoice line would be taken as long as there is some degree of similarity. The default value if nothing or an invalid entry is present is **0**. Oracle recommends setting this value to **30**. The invoice line item description must be at least 5 characters in length for the system to consider it when selecting a purchase order line item.
DescriptionDistance	Number	This value expressed as a percentage (i.e. **10** = 10%) denotes the minimum confidence distance between the best and second best possibility for the fuzzy match on the description. For example, if this value is set to 10%, and the system is 51% sure that line 1 is the right result, but 45% sure that line 2 is the right result, then the line will not pair as the distance between them (6%) is less than the minimum confidence distance. Oracle recommends setting this value to **10**.
DescriptionTolerance	Number	This denotes the percentage tolerance with which the invoice unit price is allowed to deviate from the purchase order unit price to allow the lines to pair if the pairing was via a fuzzy match on the description. If left blank, this additional check is not carried out.
UOMCheck	YES/NO	If set to **YES**, the system will take into account any possible unit of measure differences when comparing the invoice quantity to the purchase order quantity.
PUOMCheck	YES/NO	If set to **YES**, the system will apply conversions to the extracted invoice quantity based on ratios read from the purchase order if the PO unit of measure differs from the PO price unit of measure, but the invoice unit of measure and the purchase order price unit of measure are the same.
ConvertQuantityFromDB	YES/NO	If set to **YES**, the system will perform a lookup to the unit of measure conversion table specified in the *UOM section* of the project configuration if the unit of measure captured on the invoice line does not correspond to the order unit of measure on the PO line that it has been paired to, and the conversion ratio cannot be calculated mathematically based upon the PO line item details and the PO history. If the database lookup is not configured correctly, or the relevant entry is missing, the line item will not be paired.
PairToSingleGR	YES/NO	If set to **YES**, the system will only pair the invoice line to a single goods receipt line if the corresponding purchase order line is set for goods receipt based invoice verification.
FindGRWithDelivery Number	YES/NO	If set to **YES**, the system will consider the external delivery note number set against a goods receipt line in the ERP system as a priority when pairing an invoice line to a goods receipt line if the corresponding purchase order line is set for goods receipt based invoice verification.
OnlyUseDeliveryNumberTo	YES/NO	This parameter is used in conjunction with the above parameter

Parameter	Type	Description
FindGR		**FindGRWithDeliveryNumber**. If set to **YES**, the system will only pair an invoice line against a goods receipt line if the external delivery note number against the goods receipt in the ERP system can be found on the invoice. If the goods receipt in the ERP system has no external delivery note number set against it, the line will not be paired. For this feature to function, the **FindGRWithDelivery** parameter must also be set to **YES**.
RequirePODetailsFor MultipleMaterials	YES/NO	Flag to denote whether, in the event that the pool of purchase order line items contains more than one line for the same material (as denoted by an identical material number on the purchase order), the system has to find a referenced purchase order number and (optionally) a purchase order line item number on the document against the invoice line so that the system knows which purchase order line to pair the invoice line to. If this flag is set to **YES**, and no such PO details are found on the invoice, then the line item will not be paired.
IgnoreCompletedPOLines	YES/NO	If set to **YES**, during the line pairing process, the system will not consider any purchase order lines that have already been fully invoiced. 'Fully invoiced' is defined as the quantity invoiced to date being equal or greater to the quantity originally ordered.
ActivateSubDebCheck	YES/NO	If set to **YES**, the system will set the subsequent debit flag to **X** during line pairing for an invoice line that is paired to a purchase order line where the quantity that has been invoiced to date is exactly equal to the quantity that has been delivered to date and that quantity is greater than zero and the invoice unit price adjustment is less than half of the original unit price on the purchase order.
EnableIntegrityCheck	YES/NO	If set to **YES**, when carrying out line pairing against purchase orders that have multiple open lines for the same material, or have lines set for goods receipt based invoice verification where multiple open goods receipts exist, the system will not book the invoice line to the first open PO line/goods receipt. This does not apply in instances where a referenced purchase order/line at line item level, a delivery note number, or values and quantities can distinguish the correct PO line/goods receipt to book the invoice line to. This parameter should always be set to **YES** in implementations where: ▪ A workflow component sits between WebCenter Forms Recognition and the downstream ERP system ▪ WebCenter Forms Recognition is booking directly to the ERP system, but is not creating fully posted invoices. The reason for this is that the system dynamically reads the live purchase order history information from the ERP system during line pairing, so, if two invoices appear in quick succession referencing the same purchase order, both could be paired to the same PO line/goods receipt because the ERP PO history would not have been updated subsequent to processing the first invoice if it was not posted immediately. This would otherwise have made the PO line/goods receipt unavailable for the second invoice.
ExcludeFreightFromMIRA Process	YES/NO	If set to **YES**, the value of any miscellaneous charges on the invoice will not be included in a calculation to see whether there is a one-to-one relationship between the invoice and purchase order.
MultiPairingToSinglePOLine	YES/NO	If set to **YES**, the system will be able to pair more than one invoice line item to a single purchase order line item
ActivateLog	YES/NO	If set to **YES**, the system will write out a log file containing trace entries for the line pairing operation as well as for the automatic tax code determination

Parameter	Type	Description
		procedure. This is written into the standard RTS log file.
PUOMTolerance	Number	Percentage with which the PO unit price must be within the invoice unit price to infer the same order price unit of measure. If left blank, the system will look for an exact match. This is only considered if **UOMCheck** is set to **YES**. The recommended setting is **10**.
GetPOLinesFromFile	YES/NO	Flag to denote whether the purchase order line item information should be read from a flat text file.
Filename	Freetext	UNC path to the purchase order line item flat file. Each row in the file should represent a single purchase order line item with the first column being the purchase order number itself. The delimiter and column contents are optional.
ColumnSeparator	Freetext	Column delimiter within the purchase order line flat file.
ColumnNN	Freetext	Assignment of a column type code to a column in the flat file where *NN* is the column number. The available column type codes are: ▪ PO = PO Number ▪ LINE = PO Line ▪ MATERIALNO = Material Number ▪ MATERIALGROUP = Material Group ▪ DESCRIPTION = Description ▪ POQUANTITY = Quantity Ordered ▪ UNITPRICE = Unit Price ▪ POTOTAL = Order Line Total ▪ TAXCODE = Tax Code ▪ TAXJURCODE = Tax Jurisdiction Code ▪ UOM = Order Unit Of Measure ▪ PRICEUNIT = Price Unit ▪ PUOM = Order Price Unit Of Measure ▪ TOTALQUANTITYDELIVERED = Total goods receipt quantity ▪ TOTALVALUEDELIVERED = Total value of goods receipt ▪ TOTALQUANTITYINVOICED = Total quantity invoiced to date ▪ TOTALVALUEINVOICED = Total value invoiced to date ▪ ITEMCATEGORY = Item Category / Line Type (e.g. FRT) ▪ PLANT = Ship-to location code ▪ CHARGECODE = Charge Code ▪ CHARGECODEID = Charge Code ID ▪ COMPANYCODE = Company code ▪ POTYPE = JD Edwards purchase order type ▪ BUSINESSUNIT = PeopleSoft purchasing business unit ▪ ERS = PO line item ERS flag Note: **LPR_VL_COLUMN1** must be mapped to the purchase order number column.
GetPOLinesFromDB	YES/NO	Flag to denote whether the purchase order line items should be read from a database.
SQLConnectionGroup	NN	SQL connection group specifying the purchase order database connection string as set in the *SQL section*. If no connection group is specified, the system will use group **01**.
DBTableName	Freetext	Name of the database table containing the purchase order line item information.
UseStoredProcedure	YES/NO	If set to **YES**, the system will call a stored procedure in order to retrieve the purchase order line item information from the database.
StoredProcedureName	Freetext	Name of the stored procedure to be used in order to retrieve the purchase

Parameter	Type	Description
		order lines from the database
StoredProcedure Parameters	Freetext	Comma-separated list of the parameters that should be used when calling the stored procedure above. These parameters must tie in to entries in the *SPC section* of the project configuration.
DBPO	Freetext	Name of the column in the purchase order line database table that holds the purchase order number.
DBLine	Freetext	Name of the column in the purchase order line database table that holds the purchase order line number.
DBMaterialNo	Freetext	Name of the column in the purchase order line database table that holds the purchase order line item material number.
DBMaterialGroup	Freetext	Name of the column in the purchase order line database table that holds the material group to which the purchase order line item belongs.
DBDescription	Freetext	Name of the column in the purchase order line database table that holds the purchase order line item description.
DBPOQuantity	Freetext	Name of the column in the purchase order line database table that holds the order quantity for the purchase order line item.
DBUnitPrice	Freetext	Name of the column in the purchase order line database table that holds the unit price for the purchase order line item.
DBPOTotal	Freetext	Name of the column in the purchase order line database table that holds the overall total for the purchase order line item.
DBTaxCode	Freetext	Name of the column in the purchase order line database table that holds the tax code for the purchase order line item.
DBTaxJurCode	Freetext	Name of the column in the purchase order line database table that holds the tax jurisdiction code for the purchase order line item.
DBUOM	Freetext	Name of the column in the purchase order line database table that holds the order unit of measure for the purchase order line item.
DBPriceUnit	Freetext	Name of the column in the purchase order line database table that holds the purchase order line item price unit.
DBPUOM	Freetext	Name of the column in the purchase order line database table that holds the order price unit of measure for the purchase order line item.
DBTotalQuantityDelivered	Freetext	Name of the column in the purchase order line database table that holds the total quantity delivered to date for the purchase order line item.
DBTotalValueDelivered	Freetext	Name of the column in the purchase order line database table that holds the total value delivered to date for the purchase order line item.
DBTotalQuantityInvoiced	Freetext	Name of the column in the purchase order line database table that holds the total quantity invoiced to date for the purchase order line item.
DBTotalValueInvoiced	Freetext	Name of the column in the purchase order line database table that holds the total value invoiced to date for the purchase order line item.
DBItemCategory	Freetext	Name of the column in the purchase order line database table that holds the item category/line type of the purchase order line item.
DBPlant	Freetext	Location code for the ship-to address to which the goods were delivered, or where a service was performed.
DBChargeCode	Freetext	Name of the column in the purchase order line database table that holds the charge code for the purchase order line item.
DBChargeCodeID	Freetext	Name of the column in the purchase order line database table that holds the charge code ID for the purchase order line item.
DBCompanyCode	Freetext	Name of the column in the purchase order line database table that holds the company code for the purchase order line item.

Parameter	Type	Description
DBERPPOType	Freetext	Name of the column in the purchase order line database table that holds the JD Edward purchase order type for the purchase order line item.
DBBusinessUnit	Freetext	Name of the column in the purchase order line database table that holds the PeopleSoft purchasing business unit for the purchase order line item.
DBERS	Freetext	Name of the column in the purchase order line database table, which holds a flag that indicates whether this purchase order is intended for the (Evaluated Receipt Settlement (ERS) process.

4.1.28 PMT Section

This section contains settings that relate to the payment method and its relationship to the requirement for a bank account.

Parameter	Type	Description
BankMethods	Freetext	Comma-separated list of payment methods that denote a bank transfer and hence the requirement for a bank account.
AccountCurrencyNot Required	YES/NO	If this flag is set to **YES**, the system will not require a bank account currency in order to select a bank account number, however bank accounts in the vendor master that do contain a currency that matches the currency of the invoice will take priority in the bank account selection.

4.1.29 CTR Section

This section contains settings whereby the system can be pointed to a lookup table for countries to determine their local currencies and whether they are part of the EU or not. This is used in the automatic tax code determination procedure.

Parameter	Type	Description
ValidateFromDB	YES/NO	If set to **YES**, the country will be checked against a database.
SQLConnectionGroup	NN	SQL connection group specifying the country database connection string as set in the *SQL section*. If no connection group is specified, the system will use group **01**.
DBTableName	Freetext	Name of the country database table. This setting is mandatory if a database lookup is to be performed.
DBCountry	Freetext	Name of the column in the country database table that holds the country key.
DBCurrency	Freetext	Name of the column in the country database table that holds the currency for the country.
DBEUMember	Freetext	Name of the column in the country database table that denotes whether the country is an EU state member or not. **X** denotes an EU state member.
DBName	Freetext	Name of the column in the country database table that holds the full name of the country in English.

4.1.30 MAT Section

This section contains settings whereby customer material number formats are defined. At time of line item extraction, the system will strive to find a customer material number against the invoice line item. This custom material number can subsequently be used downstream in the line pairing operation as invoice lines are reconciled to their purchase order counterparts.

If no formats are defined, this does not mean that a customer material number will never be extracted; rather it increases the chance of successful recognition.

Parameter	Type	Description
NN_Format	Freetext	Format string *NN* for a possible material number. Numerous customer material number formats can be entered using incremental values (e.g. 01_Format, 02_Format, 03_Format). **#** is a wildcard character representing any number; **@** represents any alpha character The format string entered is used to help the system home in on the correct customer material number. For example, if **5######** was specified as a format string, then the system would look for a 7-digit material number beginning with **5**.
NN_Ignore	Freetext	List of characters that may appear in a customer material number in any position (e.g. a hyphen or a period/full stop) that the system should be tolerant of in the corresponding format string specified. This list does not need to be comma separated.

4.1.31 CSV Section

This section contains settings that control the CSV file output formats. The system can be configured to output multiple CSV files, either per document, or per batch.

Each group in the CSV section represents the configuration behind a single CSV file.

Parameter	Type	Description
OutputCSVFile	YES/NO	Master switch for all CSV file output. If set to **NO**, the system will not output any configured CSV files, irrespective of whether the local switch for a CSV file group is set to **YES**.
NN_OutputFile	YES/NO	Local switch for the CSV file group. If set to **YES**, the system will attempt to output a CSV file according to the configuration settings specified.
NN_CombinedFilePerBatch	YES/NO	Indicates whether one CSV file per document or one CSV file per batch is required. If output is required on a per batch basis, the output file will be created with a **.TMP** extension until the last document in the batch is exported successfully. It is not possible to output a combined file per batch unless the RTS instance carrying out the document import has a document grouping setting of either **1 folder per batch** or **1 batch per subdirectory, 1 folder per batch**.
NN_Filepath	Freetext	This parameter allows the configuration of an alternate export directory for the CSV file group. If left blank, the main export directory set against the RTS instance carrying out document export will be used; if no export directory is configured against the RTS instance, the default export directory configured in the *EXP section* of the project configuration will be used. If this is also blank, or a specified directory does not exist, the CSV file export will fail.
NN_Filename	URN [blank]	Naming convention for the CSV file if it is to be outputted on a document-by-document basis. This can be set either to **URN**, which will name the file according to the component of the image filename mapped in the *IMP section*. If left blank, or set to anything else, the filename will be set to the same name as the document filename.
NN_FileType	Freetext	File extension for the output file. If left blank, **.CSV** will be used as the file extension.
NN_FilePrefix	Freetext	File prefix for the output file.
NN_Separator	Freetext	Separator to be used in the CSV file line. Based on user input here, the system will automatically 'cleanse' any extracted data using this separator in order to maintain a consistent number

Parameter	Type	Description
		of columns in the output. Illegal separators are a full-stop/period (.), a forward slash (/) and a backslash (\). The chosen separator must still be entered manually in the format configuration for each line in the CSV file.
NN_DateFormat	DDMMYYYY MMDDYYYY YYYYMMDD	Desired output format for date fields. If no entry is made, the system will default to the output date format configured in the *EXP section* of the project configuration. If no entry is made there, the default output format of **DDMMYYYY** will be used.
NN_DateSeparator	Freetext	Separator to be used in conjunction with the date format setting above. For example, if the date format above is **MMDDYYYY** and a hyphen (-) is entered as the separator in this parameter, and the date is 2nd November 2009, then **11-02-2009** will be the output.
NN_InvoiceType	PO NPO	Filter applied to the CSV file so that output can be controlled by the value of the Invoice Type field. If this value is set to **PO**, the CSV file will only be written if the Invoice Type is **PO**. If this value is set to **NPO**, the CSV file will only be written if the Invoice Type is **NO-PO**.
NN_OutputImage	YES/NO	Selecting **YES** will output the original image document (retaining the original image filename and file type) into the same directory as the CSV file.
NN_FormatLineN	Freetext	This parameter controls the format of each row to be outputted in the CSV file. Up to five rows can be outputted per CSV file. To output the TIFF name, the invoice number and the vendor number, separated by a colon character, the setting should contain: `<%TNM>:<%INO>:<%VID>` Additional text can also be included in the format, for example: `File=<%TNM>|Invoice Number=<%INO>|Vendor ID=<%VID>` The project fields are represented by an identifier with the format <%*xxx*> where *xxx* is a three-character string representing the field. Refer to *Section 7.6.4* for a complete list of available field identifiers.
NN_LineItem	Freetext	This parameter controls the format of line item entry to be outputted in the CSV file. If left blank, no line item detail will be outputted. One row will be added into the CSV file for each line item. To output the PO number, the PO line item number and the total, this parameter should be set as follows: `<%LPO><%LPL><%LTO>` Refer to *Section 7.6.4* for a complete list of available field identifiers.

4.1.32 SPC Section

The SPC section is where parameters to be used in the stored procedure lookups are defined. Through the configuration, each parameter is given a two-digit code *NN*, which is the means by which they can be referred to in the lookups that need to employ them. Effectively, an SPC group is created for each parameter.

Each SPC group consists of five settings, which are as follows:

Parameter	Type	Description
NN_ParameterName	Freetext	Name of the stored procedure parameter. This value is mandatory for each stored procedure parameter.

Parameter	Type	Description
NN_ParameterType	Freetext	This is the stored procedure parameter type. Possible options are: ▪ BOOLEAN ▪ INT ▪ DATE ▪ DOUBLE ▪ VARCHAR For any other entries (including a blank entry), a type of **UNKNOWN** will be used.
NN_ParameterSize	Freetext	If the parameter type is **VARCHAR**, this setting specifies the maximum length allowed. If this setting is left blank, or contains either zero or a non-numeric entry, a default size of 50 will be used.
NN_ParameterValue	Freetext	This setting is where the value of the parameter is defined. There are two options for this setting: ▪ Specify the technical name of a project field, for example **PONumber**, **CompanyCode**, etc. This is case-sensitive ▪ Specify a hard value to be passed.
NN_ParameterDirection	Freetext	If set to **I**, the parameter direction will be set to input. For all other entries (including a blank entry), the parameter direction will be set to output.

4.1.33 WFR Section

This section contains configurations for the project features specific to Oracle AP processing.

Parameter	Type	Description
FilterVendorsByCompany Code	YES/NO	If this option is set to **YES**, only vendors that belong to the invoice's company code will be considered as candidates, and returned in the results of a vendor search in Verifier.
SetNegativeAmountsFor Credit	YES/NO	If this option is set to **YES** the following fields will always be exported to the XML file as negative values when the Document Type is **CREDIT**: ▪ Invoice Total ▪ Line Item Quantity ▪ Line Item Total
RemovePONumberFor NoPO	YES/NO	If this option is set to **YES**, any value in the PO Number field will be removed during document validation if the Invoice Type is **NO-PO**.
PerformDuplicateInvoice Check	YES/NO	If this option is set to **YES**, a lookup will be performed against the configured data source to determine whether the Invoice Number value already exists (i.e. whether this is a duplicate invoice). This check is performed during document validation.
CDIERPName	Freetext	The name of the ERP (or other) system that will be used in the information messages displayed as a result of the duplicate invoice check. The default value for this setting is **E-Business Suite**.
CDISQLConnectionGroup	NN	SQL connection group specifying the database connection string, as set in the *SQL section*, which will be used to perform the duplicate invoice check. If no connection group is specified the system will use group **01**.
CDIDBTableName	Freetext	The name of the database table or view containing invoice data that will be used to perform the duplicate invoice check. The default value for this setting is **XX_OFR_INVOICES_V**.
CDIDBInvoiceNumber	Freetext	The name of the field in the database table that contains the invoice number. The default value for this setting is **INVOICE_NUMBER**.

Parameter	Type	Description
CDIDBSupplierID	Freetext	The name of the field in the database tabled that contains the supplier ID. The default value for this setting is **VENDOR_ID**.
CDIDBSupplierSite	Freetext	The name of the field in the database tabled that contains the supplier site ID. The default value for this setting is **VENDOR_SITE_ID**.
CDIDBCompanyCode	Freetext	The name of the field in the database tabled that contains the company code. The default value for this setting is **ORG_ID**.
CDIIgnoreBlankValues	YES/NO	If this option is set to **YES,** the duplicate invoice check will be performed using only the configured database fields where the invoice has an extracted (or entered) value. For example, if the **CDIDBCompanyCode** setting is configured, but the document does not have a Company Code value, the lookup will be performed with that field omitted from the query.
CDIAllowForceValidation	YES/NO	If this option is set to **YES,** a message box will be displayed in Verifier stating that this is a duplicate invoice number, and the user will have the option to validate and release the document anyway. If this option is set to **NO** (default), the document cannot be released in Verifier while there is a duplicate invoice number.
EnableMetadataPassthrough	YES/NO	If this option is set to **YES,** the configured filename components will be written to the CSV and/or XML output file(s) during export.
NN_CMPFilename Component	Number	This setting identifies which component of the filename should be written to the corresponding field(s) in the CSV and/or XML files. An underscore character must be used to separate the components in the image filename. For example, assume an image filename of **AA_BB_CC_DD_EE.tif**. The value of component 4 is **DD**.
NN_CMPCSVFieldID	Freetext	The unique identifier used in the *CSV section* to determine where the component value will be written in the CSV output file. Oracle recommends that custom identifiers should begin with a **Z**, e.g. **<%ZAA>**. This setting may be left blank if it is not required for the component value to be written to the CSV file (i.e. the value is only required to be written to the XML file).
NN_CMPXMLHeader FieldName	Freetext	The tag that will be used to contain the component value in the header section of the XML output file. For example, if you want the value to appear in the XML file as **<myComponent4>DD</myComponent4>** the value for this setting would be **myComponent4**. This setting may be left blank if it is not required that the component value be written to the XML file (i.e. the value is only required to be written to the CSV file).
SPDEnableSeparator Detection	YES/NO	If set to **YES** the Separator Page Detection feature will be enabled.
SPDRequireAllPhrases	YES/NO	If set to **YES**, all of the configured separator phrases configured in NN_SPDSeparatorPhrase must appear on the same page in the document for that page to be considered as a separator page. If set to **NO** then the first page of the document that contains any of the configured separator phrases will be considered to be a separator page.
SPDDeleteSeparatorPage	YES/NO	If set to **YES** the separator page will be removed from the document in Forms Recognition.

Parameter	Type	Description
		To ensure that this modified document (with the separator page removed) is exported instead of the original image (that still contains the separator page), the following additional settings must be specified in the *EXP section* and *CSV section*: `EXP_OP_OutputTiffFile=YES` `EXP_VL_TiffName=XXX` `EXP_VL_TiffDPI=300` `EXP_VL_TiffFormat=G4FAX` `EXP_OP_RedactInvoiceNumber=NO` `CSV_OP_01_OutputImage=NO`
SPDSeparatorPhrase	NN	A phrase that signifies a separator page. Separator phrases should not include any of project's configured word segmentation characters, and other special characters such as asterisks should be avoided. Separator phrases should be a text string that is not likely to legitimately appear in an invoice document, for example, **XXX WFR SEPARATOR PAGE XXX**. Where more than one separator phrase is configured, the **NN** component must begin at **01** and run sequentially.
XCUEnableCustomUOM	YES/NO	This parameter should be set to **YES** to the *Export Custom Unit of Measure Value to XML* feature.
XCUUseISOCode	YES/NO	If this parameter is set to **YES** the value that will be written to the <*UOM*> tag for the line item in the output XML file will be the corresponding *ISOCode* for the extracted UOM value, as defined in the *UOM section* of the project configuration. For example, assume the following settings: `UOM_VL_02_ISOCode=EA` `UOM_VL_02_Alias=Each,EACH` `UOM_VL_02_ExportValue=Each` If a line item unit of measure was extracted as **EACH**, and this parameter is set to **YES**, the value written to the XML file will be **EA**, because that is the *ISOCode* setting that corresponds to the *Alias* parameter for this UOM group, where the extracted value exists in the comma-separated list of aliases. If this parameter is set to **NO** the value that will be written to the <*UOM*> tag for the line item in the output XML file will be the corresponding (optional) *ExportValue* for the extracted UOM value, as defined in the *UOM section* of the project configuration. In the example given above, the value written to the XML file would be **Each**. If the *Export Custom Unit of Measure Value to XML* feature is not enabled, or if the extracted *Alias* cannot be found in the project configuration, or if the *ExportValue* setting is not configured, the extracted unit of measure will be written to the XML file.
IFVEnableIVRForce Validation	YES/NO	If this parameter is set to **YES** the Verifier user can forcibly validate a document by selecting one of the *Invalid Reason* options defined in the *IFVInvalidReasonGroups* parameter. This allows the document to pass through to export with known invalid values in one or more of the extraction fields.
IFVInvalidReasonGroups	Freetext	Comma-separated list of Invalid Reason groups that, if selected by the Verifier user, will cause the document to be considered valid and ready for export, irrespective of whether one or more fields failed validation. For example, assume the project administrator added the following custom invalid reasons to the project configuration: `IVR_VL_10_Rule=SETAMOUNTSTOVALID` `IVR_VL_10_VerifierDisplay=COULD NOT VALIDATE DOCUMENT`

Parameter	Type	Description
		`IVR_VL_10_ExportCode=10` `IVR_VL_11_Rule=SETAMOUNTSTOVALID` `IVR_VL_11_VerifierDisplay=KNOWN INVALID DOCUMENT` `IVR_VL_11_ExportCode=11` To enable the Verifier user to select either of the above Invalid Reasons to forcibly validate a document and release it for export, this parameter should be configured as follows: `WFR_VL_IFVInvalidReasonGroups=10,11`
FXIFormatXMLforOIT	YES/NO	If set to **YES**, one or more functions of the Format XML for OIT feature will be enabled, depending on which are configured. If set to **NO**, none of the functions provided by this feature will be enabled. When this setting is set to **YES** it is important that the following settings in the *EXP section* have their values removed: `EXP_VL_XMLTableLineType=` `EXP_VL_XMLTableQuantity=` `EXP_VL_XMLTableUOM=` `EXP_VL_XMLTableUnitPrice=` Failure to remove those values will not prevent this feature from working correctly, but will result in duplicate tags being written to the XML output. Instead, the following settings in the *EXP section* must be set as follows: `EXP_VL_XMLTableOITLineType=lineType` `EXP_VL_XMLTableOITQuantity=quantity` `EXP_VL_XMLTableOITUOM=UOM` `EXP_VL_XMLTableOITUnitPrice=unitPrice`
FXIDefaultLineType	Freetext	Defines the line type value that should be used by default when the sample is configured to set blank values to the default or to replace defined values with the default.
FXISetBlanksToDefault	YES/NO	If set to **YES**, the line type tag in the XML output will be populated with the value defined in **FXIDefaultLineType**. If set to **NO** any lines where the line type is blank will have an empty tag written to the XML.
FXILineTypesToDefault	Freetext	Comma-separated list that defines which line type value(s) should be replaced with the value defined in **FXIDefaultLineType** when it is written to the XML file. This setting can be useful in implementations where the item category is being used for Service PO determination, in which case the value from the defined item category field would typically be written to the XML file as the line type. Leave this setting blank if no line type values should be set to the defined default value.
FXIEmptyQUPUOMforLine Type	Freetext	Comma-separated list that defines which line types should have empty quantity, unit price and UOM tags in the XML output file. The value(s) specified here should match the values in the line type field of the line item. Leave this setting blank if no line types should be written to the XML file with empty quantity, unit price and UOM fields (i.e. the actual values should be written to the XML file).
FXIFlipLineQUPforPOType	Freetext	Comma-separated list that defines which line types should be written to the XML file with the quantity and unit price values transposed. If enabled, the quantity value will be written to the unit price tag in the XML file, and vice versa. Leave this setting blank if this flip should not occur for any line types.

5 Customization and User Exits

5.1 Introduction

The AP Packaged Project can be customized to benefit client requirements.

Customization takes two forms:

1. Project setting customizations
2. Script customizations

Project setting customizations include such things as changing tolerances and thresholds within the project, adding new fields and classes, configuring existing ASSA fields, changing or creating Verifier forms, or setting up new users.

Important: Existing fields or classes should not be deleted or re-named under any circumstances. Doing so will invariably prove fatal to the running of the solution.

It is also possible to add script customizations into the project. For custom fields and classes, script should be added to the appropriate custom field event and class windows. For customizations to the existing **Invoices** class, this must be done in the **UserExits** class module.

5.2 AP Packaged Project Modification Restrictions

While it is expected that the AP Packaged Project will be configured and customized to meet particular business requirements, it is essential that the following restrictions are observed and adhered to when performing any kind of modifications within the project:

1. Do not modify any of the existing script code contained in the AP Packaged Project
   - All custom scripts must be written in the **UserExits** module
   - For newly created fields on the **Invoices** class, you are permitted to add custom validations
   - You are permitted to add functions/subroutines at the end of the **GlobalVariables** module
2. Do not modify the generic learnset provided with the AP Packaged Project
   - The only exception to modifying the learnset is to add additional documents for the sole purpose of training new fields
   - If a customer is experiencing a problem with a vendor's invoice, you can use Supervised Learning Workflow (SLW) to create a vendor subclass to maintain vendor-specific training
3. Do not re-train any of the currently trained fields within the AP Packaged Project
4. Visual changes on the Verifier form are permitted

5.3 Script Customizations

5.3.1 Organization of Project Script

The application script is organized logically across several classes. These classes, along with their associated scripts, are described in the sections below.

5.3.1.1 Project Script Class

The project script class contains script associated with standard AP Packaged Project system events, which are known as the *ScriptModule* events. These *ScriptModule* events are called at specific points within the project workflow.

For example: pre and post-import, pre and post-OCR, pre and post-classification, and at time of document export.

A developer seeking to customize the project should not make any changes to the code on this class level. Doing so may prove fatal to the running of the solution.

5.3.1.2 GlobalVariables Script Class

The *GlobalVariables* script class houses all global variables that are used within the AP Packaged Project. These data definitions are exposed so that a project developer can elect to use them within custom code if appropriate, and also so that the developer can see the definition of the custom structures and arrays as a point of reference.

In addition to global variables, this script class also houses a series of common functions and subroutines used throughout the solution. The developer has the option of using these common functions within custom code placed in the *UserExits* script class, or within code for any additional classes that are created.

The common functions, along with a description of their potential uses, are described in the table below:

Function / Subroutine	Description
ReadPrjINI	This function reads the project configuration (.ini) file.
DicVal	This function returns the value of any parameter contained within the project configuration (.ini) file. The function parameters are as follows: ▪ strKey = the name of the project configuration parameter. ▪ strDic = the name of the .ini section in which the parameter is held. Neither *strKey* nor *strDic* are case-sensitive. Example usage: If project configuration parameter **EXP_VL_OutputDateFormat** contains **MMDDYYYY**, then the following command will populate the local string variable *strOutputDateFormat* with **MMDDYYYY**: `strOutputDateFormat = DicVal("OutputDateFormat", "EXP")` For project configuration parameters that have a Boolean type of **OP**, the function will only return a value of **YES** or **NO**.
Parse_INIVal_Yes	This function receives a value (*strVal*) that has been entered against a parameter in the project configuration (.ini) file with Boolean type **OP**, and determines whether the value should be interpreted as **YES** or **NO**.
SplitString	This function performs a split on a given string, based on a nominated separator, and returns the components of the string back to the calling function in an array, along with

Function / Subroutine	Description
	the number of values in the array. The interface parameters are as follows: ▪ strSource = input string to be split ▪ strSplitArray = array containing the split results passed back to the calling module ▪ strDesignator = the delimiter to be used when performing the split ▪ ArrayLineCount = the number of array elements in the returned strSplitArray Example usage: `Dim myString  As String` `Dim Words()  As String` `Dim intWordCount As Integer` `myString = "MARY HAD A LITTLE LAMB"` `' A space is set as the delimiter` `Call SplitString(myString, Words(), " ", intWordCount)` The returned *Words* array would contain the following: ▪ Words(1) = "MARY" ▪ Words(2) = "HAD" ▪ Words(3) = "A" ▪ Words(4) = "LITTLE" ▪ Words(5) = "LAMB" The returned *intWordCount* parameter would be set to **5**.
fnConvertToExternal	This function will convert a date in the format used internally within the AP Packaged Project (i.e. DD/MM/YYYY) into a specified format. The interface parameters are as follows: ▪ strDate = date to be formatted ▪ strFormat = format of *strDate* (either **MMDDYYYY** or **YYYYMMDD**. Any other entry will return **DDMMYYYY**) ▪ strSeparator = separator to be used when converting the date Example usage: `Dim myDate as string` `myDate = "12/08/2009"  ' 12 August 2009` `myDate = fnConvertToExternal(myDate, "MMDDYYYY", "-")` The value of *myDate* will now be set as **08-12-2009**.
fnConvertToInternal	This function is used to convert a date with a specified format into the date format used internally within the AP Packaged Project (i.e. DD/MM/YYYY). The interface is as follows: ▪ strDate = date to be formatted to DD/MM/YYYY ▪ strFormat = current format of *strDate* (either **YYYYMMDD** or **MMDDYYYY**. Any other entry will return **DDMMYYYY)** ▪ strSeparator = separator currently used in *strDate* Example usage: `Dim myDate as String` `myDate = "2009-08-12"  ' 12 August 2009` `myDate = fnConvertToInternal(myDate, "YYYYMMDD", "-")` The value of *myDate* will now be set to **2009-08-12**.
fnFormatDateForExport	This function converts a date in the Verifier output format, as configured in the *DAT section* of the project configuration, into a date in the export output format, as configured in the *EXP section*.

Function / Subroutine	Description
	For example, an invoice date can potentially appear in any format, but the system will convert it to the format specified in the *DAT section*. If that format is **MMDDYYYY**, then 12th August 2009 will be displayed in Verifier as **08/12/2009**, which is also the technical content of the field object text property. *fnFormatDateForExport* takes the technical contents of the field, and converts it into the date format as specified in the *EXP section*. So, if the export format is **YYYYMMDD** with a hyphen as the separator, then the following command will populate string variable *strDate* with **2009-08-12**: `strDate = fnFormatDateForExport(pWorkdoc.Fields("MyDate").Text)` The interface of the function is as follows: ▪ strDate = date to be converted
fnWriteXMLField	This function writes a single line into the XML file, and is intended for use within *UserExitXMLOutput* to provide a developer with a mechanism to add a custom field into the XML file with just a single command. The interface of the function is as follows: ▪ strAttribute = name of the project configuration parameter containing the tag for the XML field ▪ strValue = value of the field to be outputted Example usage: In the project configuration, a new parameter has been created **EXP_VL_XMLHCInvoiceCode** with a value of **invCode**. A new field has been created against the invoices class in the project called **InvoiceCode** that contains the extracted value of **12345**, and this value should be written to the invoice header section of the XML file. This can be achieved by putting the following in the *UserExitXMLOutput* function: `Select Case strSection` `Case cXMLDocHeader` `…` `Case cXMLInvHeader` `fnWriteXMLField("HCInvoiceCode",pWorkdoc.Fields("InvoiceCode")` `.Text)` `End Select` This will write out the following line into the invoice header section of the XML file: `<invCode>12345</invCode>`
fnWriteXMLDateField	This function is used to write out a date field to the XML file where the date to be written is in the Verifier output date format specified in the *DAT section* of the project configuration. As well as writing the value into the XML file, the system will convert the date passed into the date export format as specified in the *EXP section* of the project configuration. The interface and function usage is identical to that of *fnWriteXMLField* described above.
fnWriteDBHeaderField	This function is used to write an additional database header field into a downstream database if database export is activated in the *EXP section* of the project configuration. It is intended to provide a developer with a single command to accomplish this within *UserExitDBHeaderExport*. The interface of the function is as follows: ▪ strAttribute = name of the project configuration parameter containing the table column mapping for the destination field in the database. ▪ strValue = field value to be written to the database. ▪ strINIFileKeyName = current project configuration parameter being assessed by the database header output routine. This value should be passed unaltered from the value passed into the user exit.

Function / Subroutine	Description
	▪ strFieldValue = current value of the field to be written into the database header table. This value should be passed unaltered from the value passed into the user exit Example usage: In the project configuration, a new parameter has been created called **EXP_VL_DBHCInvoiceCode** with a value of **INVCODE**, which denotes the technical column name of the downstream invoice header database table. A new field has been created against the *Invoices* class in the project called **InvoiceCode** that contains the extracted value of **12345**. To write the field value into column **INVCODE** in the database document header table, insert the following code into *UserExitDBHeaderExport*: `fnWriteDBHeaderField("InvoiceCode", pWorkdoc.Fields("InvoiceCode").Text, strINIFileKeyName, strFieldValue)` When the export runs, the database column **INVCODE** will now be populated with the value **12345**.
fnWriteDBHeaderDateField	This function is used to write out a date field to the invoice header database table where the date to be written is in the Verifier output date format specified in the *DAT section* of the project configuration. As well as writing the value into the database table, the system will convert the date passed into the date export format as specified in the *EXP section* of the project configuration. The interface and function usage is identical to that of *fnWriteDBHeaderField* described above.
fnGetFileName	This function receives a full filename, which includes the file path and file extension, and returns the name of the file itself, excluding the path and extension. For example, if **c:\My Documents\MyImage.tif** is passed to the function, the output will be **MyImage**. Interface: strFileName
fnGetBaseClass	This function returns the name of the base class associated with the class passed to the function. If a base class is passed to the function, the same base class is returned. For example, if the function receives **ExpenseSheets** and that class is a subclass of the *Invoices* base class, the function will return **Invoices**. Interface: strClass
fnIsVerifier	This function returns a Boolean **TRUE** value if the current WebCenter Forms Recognition module that is executing the script is the Verifier application.
fnGetBatchID	This function receives the path to a document in the batch directory, parses the file path, and returns the batch ID number as a string. Interface: strWorkfile
fnIsAlpha	This function returns a Boolean **TRUE** value if the string passed via the parameter *strString* is made up entirely of alpha characters. Interface: strString
fnGetUserDecimalSeparator	This function reads the local Windows settings for the user logged onto the machine and returns either a full stop/period, or a comma depending on the decimal separator preferences.
fnWriteCSVField	This function replaces a user-defined literal in the CSV file configuration with its corresponding value, and is intended for use within *UserExitCSVFile* to provide a developer with a mechanism to add a custom field into the CSV output file with just a single command. The interface of the function is as follows: ▪ strRecordtext = current text of line to be written into the CSV file

Function / Subroutine	Description
	▪ strKey = CSV file group number ▪ strSymbol = user-defined literal to be replaced ▪ strValue = value to replace the literal with Example usage: If CSV group **01** has the parameter **CSV_VL_01_LineFormat1=<%ZIC>,** where **<%ZIC>** is intended to represent the extracted value of custom field *InvoiceCode*, then the function should be called as follows in *UserExitCSVFile*: `fnWriteCSVField(strRecordText, strKey, "<%ZIC>", pWorkdoc.Fields("InvoiceCode").Text)`
fnWriteCSVDateField	This function replaces a user-defined literal in the CSV file configuration with its corresponding date value, and is intended for use within *UserExitCSVFile* to provide a developer with a mechanism to add a custom field into the CSV output file with just a single command. This function should only be used if the date value to be written into the CSV file is in the Verifier output format configured in the *DAT section* of the project configuration. The date will be outputted in the date format configured for the CSV file group. If no configuration has been made here, it will be formatted according to the date output format configured in the *EXP section* of the project configuration. The interface and function usage is identical to that of *fnWriteCSVField* described above.
fnSetDBConnection	This function can be called from a user exit in order to connect to a database. The function takes in a database connection string via input parameter *strConnection*. If the connection is already available, the index of the connection in global database connection array *objDBConn* will be returned. If it is not available or not open, the function will initialize the connection and return the relevant index of the *objDBConn* object. If the connection cannot be made, the function will return a value of **-1** and an appropriate error message will be written into the standard project log file. Example usage: This following code will instantiate a database connection and execute a SQL call where variable *myDBConnection* represent the connection string, and *mySQL* represents the SQL statement: `Dim lngConnection As Long` `Dim myConnection As ADODB.Connection` `lngConnection = fnSetDBConnection(myDBConnection)` `If lngConnection = -1 Then` `  ' Connection could not be made – error handling` `Else` `  ' Execute SQL using connection` `  Set myConnection = objDBConn(lngConnection)` `  myConnection.Execute(mySQL)` `End If` *objDBConn* is a global database object available for use in any user exit. Interface: strDBConnection
fnMatchDBComponents	This is a supporting function used by the function *fnSetDBConnection*.
fnCheckDBArray	This function checks whether a passed database connection array of type *ADODB.Connection* is initialized. If it is not, the function will then initialize it. Interface: myArray()
fnExtractDBComponents	This is a supporting function used by *fnSetDBConnection*.
fnGetFieldAnalysisSettings	This function returns an instantiated *AnalysisSettings* object for given associative search

Function / Subroutine	Description
	engine field *oASSA* and document class *strClass*. Interface: strClass, oASSA
fnIsValueInList	This function takes a comma-separated list in the input parameter *strList* and a string value *strValuePreserve*. The function will return a Boolean **TRUE** value if the value in the varieable *strValuePreserve* is one of the values in the list. Interface: strList, strValuePreserve
fnConvertToDouble	This function converts the value of the passed *strString* variable to a double value in a way that is consistent with the locale settings of the machine. If the string cannot be converted, the output will be zero. Interface: strString
fnIsNumeric	This function returns a Boolean **TRUE** value if all characters passed in the input parameter *strTemp* are numeric. Interface: strTemp
fnGetMiscChargeTotalForCode	This function adds up all of the invoice miscellaneous charges for a given miscellaneous charge code, *strCode* as registered in the *MSC section* of the project configuration, and returns this total via the function name. Interface: pWorkdoc, strCode
fnCalculateExchangeRate	This function calculates the correct exchange rate to pass during document export, based upon user entry in the exchange rate and local VAT amount fields in the Verifier application. The exchange rate is passed out as a string via the function module name. Example usage: `Dim strExchRateExport As String` `strExchRateExport = fnCalculateExchangeRate(pWorkdoc)` Interface: pWorkdoc
fnCheckForNull	This function receives a field component of a database record set and returns the value as a string to the calling routine via the function name. If the field component has a null value, an empty string is returned. Interface: strString

5.3.1.3 UserExits Script Class

This class contains the project user exit script points.

The developer should not remove or change the definitions of the user exits provided. Doing so will prove fatal to the running of the solution.

5.3.1.4 Invoices Script Class

The *Invoices* script class contains the source code for the invoice class validation events, which includes the logic that is used to validate fields and the document as a whole, as well as to control the behavior of the Verifier form.

A developer customizing the solution is free to add new validation events that correspond to newly created fields on the Invoices class. These extra events should be created at the end of the existing code in the area marked in the script.

No changes should ever be made to any of the existing Oracle-delivered code. Doing so may prove fatal to the running of the solution.

5.3.1.5 APPackaged and Generic Classes

These classes are for Oracle internal use only. They should not be deleted, changed or renamed. The consequences of doing so will be fatal to the running of the solution and the project file may not be recoverable.

5.3.1.6 Sequence of Class Dependencies

When making changes to the script, the developer should be mindful that dependencies exist between the various script layers, so it is not possible to run one script if there is a dependency on a script which is not already running. Attempting to run a script will also perform a syntax check.

For that reason, the scripts must be started in sequence. For the AP Packaged Project, the required sequence is:

1. GlobalVariables
2. UserExits
3. Project

5.3.2 User Exits

A user exit is a dedicated public subroutine or function on the *UserExits* class script level where custom code may be inserted by project developers.

Each user exit is called from a relevant point in the application layer baseline code and provides the developer with a 'hook' to perform a custom activity as is appropriate for a client implementation.

Customizations carried out by project developers should be implemented in a modular fashion within the user exits made available. If any ancillary functions are required to support these modules, then these should be created as public functions. These ancillary functions can be placed on the *UserExits* script class if they are only to be used locally; if they need to be accessed by custom script on other classes, they can be placed at the end of the existing script on the *GlobalVariables* class in the marked area.

The user exits currently available, along with their calling points and suggested uses, can be found in the table below:

User Exit	Calling Routine(s)	Description
UserExitCustomExport	ScriptModule_ ExportDocument	This is the user exit for custom export modules, for example, custom flat files or custom database updates. This is the only user exit that has a corresponding activation parameter within the *EXP section* of the project configuration. This is in keeping with all other export output options. Interface: pWorkdoc, ExportPath, strDocLink, LineData, GLData, TaxData, blLinesRequired, Address, Flags The global variable *strExportError* should be populated with an appropriate error message if the export fails. This will have the effect of setting the batch to a status of 750, with the error message set against the invoice number. This user exit will only be called for documents that have not been voided. Special handling for voided documents should be inserted in the user exit *UserExitVoidDocumentExport*.
UserExitPostExtract	Document_ PostEvaluate	This is the user exit used to set any custom field defaults, or to re-evaluate any extracted fields.

User Exit	Calling Routine(s)	Description
	(Invoices class)	Interface: pWorkdoc
UserExitRouteDocument	ScriptModule_ RouteDocument	This is the user exit for performing any custom activity connected to the project workflow state of each document, i.e. changing the state based on a property of the workdoc or document filename so that they can be filtered on a user-by-user basis. Interface: pWorkdoc, State
UserExitPONumberPostEvaluate	PONumber_ PostEvaluate (Invoices class)	This is the user exit where a custom routine can be added to re-evaluate the weightings for candidates for the Purchase Order number field, for example, to check for their existence in an external database. Interface: pField, pWorkdoc
UserExitVoidDocumentExport	ScriptModule_ ExportDocument	This is the user exit provided for the custom export of documents belonging to the *Void* class. Interface: pWorkdoc, ExportPath, strDocLink The global variable *strExportError* should be populated with an appropriate error message if the export fails. This will have the effect of setting the batch to a status of 750, with the error message set against the invoice number.
UserExitPONumberValidate	PONumber_Validate (Invoices class)	This user exit can be used for custom purchase order number validations. It is called subsequent to the purchase order number being validated against a database table. Interface: pField, pWorkdoc, pValid, lngIndex If the PO number valid flag is to be changed, then the values of *pValid* and *pWorkdoc.Fields("PONumber").Valid* should be changed to the new Boolean value. The PO header and line details are passed in global arrays *g_POHeader* and *g_POLines*. The *lngIndex* parameter contains the index number of the purchase order record in the *g_POHeader* array. The definition of the arrays can be found on the *GlobalVariables* script level. If the purchase order details are being read from a database, the line item array will not be populated if line pairing is switched off.
UserExitPOValidateStart	PONumber_Validate (Invoices class)	This is the user exit used for custom purchase order number validations to be undertaken at the start of the PO number validate routine. Interface: pField, pWorkdoc, pValid, blExit If the PO number valid flag is to be changed, then the values of *pValid* and *pWorkdoc.Fields("PONumber").Valid* should be changed to the new Boolean value. If the parameter *blExit* is set to **TRUE**, the *PONumber_Validate* routine will exit immediately after returning from the user exit. This user exit is superseded by *UserExitPOValidateStart2*.
UserExitTerminate	ScriptModule_ Terminate	This user exit is called from the beginning of *ScriptModule_Terminate*. It can be used to unload any global script objects employed in custom script. Interface: ModuleName

User Exit	Calling Routine(s)	Description
UserExitPreImport	ScriptModule_ PreImport	This user exit is called from the beginning of *ScriptModule_PreImport*. Interface: pWorkdoc, FilePath, FileType, pCancel
UserExitPostClassify	ScriptModule_ PostClassify	This user exit is called from the beginning of *ScriptModule_PostClassify*. Interface: pWorkdoc
UserExitDocumentTypeValidate	DocumentType_ Validate (Invoices class)	This user exit is called at the beginning of *DocumentType_Validate* on the Invoices class. It can be used to set the valid flag of the document type depending on whether it is an invoice or credit note. Interface: pField, pWorkdoc, pValid
UserExitAmountMiscPostEvaluate	Internal application	This user exit can be used to evaluate the weighting of candidates for a miscellaneous charge in the AmountMisc field in a manner that is appropriate for the project implementation where the desired contents of the field can change depending on client requirements. Interface: pField, pWorkdoc
UserExitChangeReportingCountry	Internal application	This user exit permits a change in the reporting country for countries that do not use tax jurisdictions. This exit is implemented as a function, and, under no circumstances should the default line below be changed without something appropriate being in its place: `UserExitChangeReportingCountry = strCountry` The parameter *strCountry* is initially set to the country of the company code to which the invoice is to be posted, which is assumed to be the tax reporting country. As some companies have sites abroad that are VAT registered in that country, the company code country is not appropriate as a key to retrieve the appropriate tax codes from the project's Tax Table. This user exit provides an opportunity to implement specific business logic to select the correct country, which, in the example above, would be the country where the plant is located. For PO invoices, the plant can be retrieved from the POLines array in the **werks** property. Interface: strCountry, pWorkdoc
UserExitSetTolerance	Internal application	This user exit allows greater flexibility for setting the tolerance against which amount fields are cross-validated mathematically within the project. Standard configuration permits different tolerance values to be assigned to different currencies, but if a further dimension is required (e.g. to consider the company code), then this logic should be implemented within this user exit. Adjusting the tolerances requires manipulating the *Tolerance* array. Interface: Tolerance, pWorkdoc
UserExitDocumentOnAction	Document_OnAction (Invoices class)	This user exit provides an opportunity for a developer to add script that relates to custom buttons that they may elect to add to the Verifier form. The *ActionName* parameter, which is passed into the function, is populated with the technical name of the action associated with a user pressing the button as designated in

User Exit	Calling Routine(s)	Description
		Verifier Design Mode in the Designer application. Interface: pWorkdoc, ActionName
UserExitDBHeaderExport	Internal application	This user exit permits a developer to add custom header fields into the standard database export function, or to change the value of an existing field. A corresponding project configuration parameter (EXP_VL_DBHCxxx) must be added to the project configuration to designate the mapping between the export field and the column name of the invoice header database table into which the field should be written. Interface: strINIFileKeyName, pWorkdoc, strDocLink, strFieldValue *strINIFileKeyName* denotes the name of the field as per the project configuration parameter. For example, if the parameter is **EXP_VL_DBHCMyField**, then *strINIFileKeyName* will be set to **MyField**. *strFieldValue* is the value to be exported to the database. *strDocLink* is the link to the image of the document, and could be a filepath or a URL, depending on settings in the *REP section* of the project configuration.
UserExitXMLOutput	Internal application	This user exit is available for a developer to add any custom fields into the XML output file. The XML output file is divided into four sections: ▪ The document attributes (e.g. class name, scan date); ▪ The invoice header (e.g. invoice number, date); ▪ The line item detail (e.g. quantity, unit price) ▪ General ledger coding (e.g. GL account, cost center) Custom fields can be entered into any of these four sections by use of the public *fnWriteXMLField* and *fnWriteXMLDateField* functions. This user exit is called by the core application code once for the document header, once for the invoice header, once for each standard line item and once for each general ledger coding line item. The *LineData* array parameter contains the standard line items; the *GLData* array parameter contains the general ledger coding lines. The formal parameter *strSection* denotes the section of the XML file that the exit is being called for. *lngLine* denotes the index of the line item that the user exit is being called for. Interface: pWorkdoc, LineData(), GLData(), lngLine, strSection
UserExitCSVFile	Internal application	This user exit is called for each header line outputted to the CSV file, and can be used to map custom user literals to their desired value counterparts; for example, to include a custom field in the CSV file output. Functions *fnWriteCSVField* and *fnWriteCSVDateField* are provided for this purpose to condense the operation into a single command. The parameter *strRecordText* contains the text of the current line to be written into the file. *strKey* is the CSV index group number for the file being processed. Interface: pWorkdoc, strRecordText, strKey

User Exit	Calling Routine(s)	Description
UserExitCSVFileLine	Internal application	This user exit is called for each line item outputted to the CSV file, and can be used to map custom user literals to their desired value counterparts, for example, to include a custom line item component in the CSV file output. Functions *fnWriteCSVField* and *fnWriteCSVDateField* are provided for this purpose to condense the operation into a single command. The parameter *strRecordText* contains the text of the current line to be written into the file. *strKey* is the CSV group index number for the file being processed. The *LineData* structure is also passed in containing details of the current line being outputted. Interface: pWorkdoc, LineData, strRecordText, strKey
UserExitExportSuccess	ScriptModule_ ExportDocument	This user exit is called at the point where it is known that all selected exports have been successful for the document being processed. It can be used to update additional reporting data if required. Interface: pWorkdoc
UserExitExportFailure	ScriptModule_ ExportDocument	This user exit is called at the point where it is known that export has failed for the document being processed. It can be used to update additional reporting data if required. The reason for the export failure can be found in the global parameter *strExportError*. Interface: pWorkdoc
UserExitValueCheck	Internal application	This user exit is called from the Invoice Number and Invoice Date *PostEvaluate* events for documents that are classified to the **Invoices** class. It is called once per candidate, and is provided as a window for a developer to insert script to disqualify illegal candidates for the invoice number or invoice date. Interface: pField, pWorkdoc, oCandidate
UserExitLinePairingPOs	Internal application	This user exit provides functionality to edit the list of purchase orders that are to be considered during the line pairing operation. New purchase orders may also be added based on criteria that may be coded within the user exit. Interface: pWorkdoc, PO() The *PO* parameter contains the details of the purchase orders to be used during line pairing. It is based on the *POKey structure*. If an error occurs during the user exit code, then global variable *strExportError* should be populated with the reason for the error. Populating this variable will have the effect of failing the document export.
UserExitVerifierException	ScriptModule_ VerifierException	This user exit is triggered when a user sends a document to an exception state in Verifier. Interface: pWorkdoc, Reason, CreateNewBatch, BatchName, BatchDocumentState, BatchPriority, BatchFolderName, ApplyExceptionHandling
UserExitFilterVendorSearch	Document_Post Extract	This user exit allows a developer to filter the list of vendors shown in the Verifier vendor search facility, and also to

User Exit	Calling Routine(s)	Description
	Internal application	change the information that is displayed. It can also be used to remove vendors that the system should not consider to be the correct vendor, or to adjust the confidence weightings of candidates that are under consideration to be the correct vendor. Interface: pWorkdoc, oCandidate, Address, strDisplay, blReject The *oCandidate* parameter contains the vendor candidate object, and the *Address* parameter contains the vendor *Address data structure*. The *strDisplay* parameter contains the current vendor search box display line for the vendor. This will be blank if the user exit is called prior to the initial determination of the correct vendor on server side. Checking whether this string is empty or not allows a different behavior to be defined between server side and search box functions. The *blReject* parameter is initially set to **FALSE**, but the developer should set it to **TRUE** if the vendor should be excluded from the search results or from the extraction results.
UserExitSetReportingLoginName	Internal application	This user exit allows a developer to change the name of the user as reported in the reporting database. This should be used in implementations where the Web Verifier is being used with versions of the AP Packaged Project earlier than 1007D; otherwise, the system will always populate the Verifier user column in the reporting database with the Forms Recognition service user. Input parameter *strUserName* contains the user name that the system is currently using. Interface: pWorkdoc, strUserName
UserExitPreMaterialLinePairing	Internal application	This user exit is called prior to the commencement of material line pairing. It is not called for the pairing of service line items. The *LineData* input parameter will always be empty at this point. *dblPOTotal* contains the total value of all purchase orders being considered during line pairing. *dblGRTotal* contains the total outstanding value of all purchase order numbers being considered during line pairing (i.e. value of total goods receipt against all purchase orders minus the value of total invoice receipt against all purchase orders). This user exit can be used to skip the MIRA process of line pairing by setting both *dblPOTotal* and *dblGRTotal* to zero, although this is not recommended by Oracle. Interface: pWorkdoc, LineData(), dblPOTotal, dblGRTotal
UserExitPostLinePairing	Internal application	This user exit is called after the line pairing and automatic tax determination functions have been completed during document export, but before any data outputs have actually been carried out. It gives the developer an opportunity to look at the line pairing results (held in the *LineData* array) and make any changes or additional customizations as required. Interface parameter *blLinesRequired* will be set to **TRUE** if line

User Exit	Calling Routine(s)	Description
		items are to be exported for the document in question. Interface: pWorkdoc, LineData(), TaxData(), blLinesRequired
UserExitAddressArray	Internal application	This user exit is called each time the details for a vendor are read from the vendor pool. It gives the developer an opportunity to amend or add new parameters to the *Address*. The user exit will not be called if the vendor details have already been read and loaded into the local cache. Interface: oASSA, strVendorID, Address
UserExitDocumentValidate	Document_Validate (Invoices class)	This user exit can be used to code additional document-level validations and activities. Interface: pWorkdoc, pValid
UserExitEditDocumentWeblink	Internal application	This user exit permits a developer to manipulate the document web link, as stored in the project reporting database, and exported downstream. The current web link is passed into the user exit via interface parameter *strWebLink*, and this may be changed as required. The current unique document ID is passed in via interface parameter *strDocID*, which may not be changed. Interface: strWebLink, strDocID
UserExitInvoiceNumberValidate	InvoiceNumber_ Validate (Invoices class)	This user exit is available to include additional validations and formatting against the invoice number field. Interface: pField, pWorkdoc, pValid
UserExitPOValidateStart2	PONumber_Validate (Invoices class)	This user exit is called immediately after *UserExitPOValidateStart*, and is intended for the same functional purpose, except that it has an extended interface to make it easier to use and work with than *UserExitPOValidateStart*. Included in the interface is the *Address structure*, which contains the full details of the current vendor. Also included is the *POKey structure*, the contents of which may be changed if required. This was not previously possible with *UserExitPOValidateStart*. Interface: pField, pWorkdoc, pValid, blExit, POKey, Address
UserExitSetDocFlags	Internal application	This user exit allows a developer to set the properties on the *Flags* object based on a custom set of requirements. Interface: pWorkdoc, Flags
UserExitInvoiceDateValidate	InvoiceDate_Validate (Invoices class)	This user exit is available for custom validation logic to be added to the Invoice Date field validation event. Interface: pField, pWorkdoc, pValid
UserExitDueDateValidate	DueDate_Validate (Invoices Class)	This user exit is available for custom validation logic to be added to the Invoice Due Date field validation event. Interface: pField, pWorkdoc, pValid
UserExitDeliveryDateValidate	DeliveryDate_ Validate (Invoices class)	This user exit is available for custom validation logic to be added to the Delivery Date field validation event. Interface: pField, pWorkdoc, pValid
UserExitVerifierFormLoad	ScriptModule_ VerifierFormLoad	This user exit is called after the Verifier form is loaded, at the end of the *ScriptModule_VerifierFormLoad* event.

User Exit	Calling Routine(s)	Description
		Interface: pWorkdoc, FormClassName, FormName
UserExitScriptModuleInitialize	ScriptModule_ Initialize	This user exit is called at the end of the *ScriptModule_Initialize* event. Interface: pWorkdoc
UserExitPostImport	ScriptModule_ PostImport	This user exit is called from the beginning of the *ScriptModule_PostImport* event. Interface: pWorkdoc
UserExitPostImportBatch	ScriptModule_ PostImportBatch	This user exit is called at the beginning of the *ScriptModule_PostImportBatch* event. Interface: pWorkdoc
UserExitPreClassify	ScriptModule_ PreClassify	This user exit is called from the beginning of the *ScriptModule_PreClassify* event. Interface: pWorkdoc
UserExitUpdateSystemSecurity	ScriptModule_ UpdateSystem Security	This user exit that is called during the system security update event, which is set to run as a periodic background job on the runtime server. Interface: InstanceName
UserExitMoveDocument	ScriptModule_ MoveDocument	This user exit is called when a document is sent to an exception batch in Verifier. The internal application uses this event to update the reporting tables with any change in the document batch ID. Interface: pWorkdoc, OldBatchID, NewBatchID, Reason
UserExitBatchOpen	ScriptModule_ BatchOpen	This user exit is called when a batch is opened in Verifier. Interface: UserName, BatchDatabaseID, ExternalGroupID, ExternalBatchID, TransactionID, WorkflowType, BatchState
UserExitProcessBatch	ScriptModule_ ProcessBatch	This user exit is called during the *Custom Processing* workflow step. Interface: pBatch, InputState, DesiredOutputStateSucceeded, DesiredOutputStateFailed
UserExitBatchClose	ScriptModule_ BatchClose	This user exit is called when a batch is closed in Verifier. Interface: Username, BatchDatabaseID, ExternalGroupID, ExternalBatchID, TransactionID, WorkflowType, BatchState
UserExitAppendWorkdoc	ScriptModule_ AppendWorkdoc	This user exit is called when a user merges documents together in Verifier. Interface: pLastWorkdoc, pCurrentWorkdoc, pAppendType
UserExitPreClassifyAnalysis	ScriptModule_ PreClassifyAnalysis	This user exit is called during the classification event where the classification matrix may be adjusted or extended to influence the classification result. Interface: pWorkdoc
UserExitCompanyCodeValidate	CompanyCode_ Validate (Invoices class)	This user exit is available for custom validation logic to be added to the Company Code field validation event. The interface is pField, pWorkdoc, pValid.
UserExitFocusChanged	Document_ FocusChanged (Invoices Class)	This user exit is called during the standard *FocusChanged* event in Verifier. This event is called each time the field or table cell focus in Verifier is changed, or when the system moves to a new document.

User Exit	Calling Routine(s)	Description
		Any script inserted into this user exit should be kept to a minimum as overly time-consuming operations could make the Verifier application cumbersome to use. For example, script that is relevant only when the system moves on to a new document would be better placed in *UserExitVerifierFormLoad*. The interface is pWorkdoc, Reason, OldFieldIndex, NewFieldIndex
UserExitCheckBankAccount	VendorID_Validate PONumber_Validate (Invoices class)	This user exit is called during the bank account check, where the system attempts to determine the appropriate bank account based upon the content of the document and the bank records present in the vendor master data. It allows a developer to customize alternative logic for the bank account selection. The import parameter *Address* contains the details for the currency vendor, and the bank account records specified against the *BankDetails* property. The invoice currency is passed into the user exit via the *strCurrency* parameter. The interface is pWorkdoc, Address, strCurrency

5.3.3 Custom Error Messages

If script code placed within the user exit framework is to include custom error messages, then these may be included as entries in the *ERR section* of the project configuration, rather than being hard-coded within the script. The error message number range assigned for customer usage is 900-999, which should be adhered to in order to prevent any conflicts in the event of an upgrade.

For example, in the project configuration file, the following parameter has been added:

```
ERR_VL_900=Please check data entry
```

This can be retrieved through the script with the following code:

```
Dim myError As String

myError = DicVal("900", "ERR")
```

Hence, local string *myError* will now contain **Please check data entry**.

5.3.4 Project Data Structures

The AP Packaged Project uses internal data structures to pass data between functions and subroutines. It is possible to use some of these data structures in user exit script, and, in some cases, these structures are defined a formal parameters in the interface. The structures available are defined below.

5.3.4.1 LineData Structure

The *LineData* structure contains the line item detail that is going to be exported. Fundamentally, it is based on the document line item extraction results, but may deviate from that depending on the results of line pairing (for example, if a single line on the invoice corresponds to more than one line item on the purchase order) and also based upon how invoice miscellaneous charges are handled.

The LineData structure is present in the interfaces to the following user exits:

- UserExitCustomExport
- UserExitXMLOutput
- UserExitCSVFileLine
- UserExitPreMaterialLinePairing
- UserExitPostLinePairing

The elements contained within the structure are as follows:

Structure Element	Type	Description
INVOICE_DOC_ITEM	Integer	Invoice line item number from 1-n
PO_NUMBER	String	Purchase order number. This is populated only if line pairing has been successful for this item.
PO_ITEM	String	Purchase order line item number. This is populated only if line pairing has been successful for this item.
DE_CRE_IND	String	Subsequent debit/credit indicator. This denotes whether the line item is a subsequent debit of a credit line item. If this value is set to **X** and the document type is **INVOICE**, then the line item is treated as a subsequent debit (amount only, not quantity). If the value is set to **X** and the document type is **CREDIT**, then the line item is treated as a subsequent credit. If the value is blank, the line item is treated as a regular line item.
QUANTITY	Double	Invoice line item quantity.
ITEM_AMOUNT	Double	Invoice line item total.
PO_UNIT	String	Order unit of measure. This will be populated with the purchase order line item order unit of measure if the line item has been paired.
PO_PR_UOM	String	Order price unit of measure. This is populated with the order price unit of measure from the purchase order line item if the line item has been paired. In all other cases, it will be blank.
PO_PR_QNT	Double	Invoice line item quantity expressed in the order price unit of measure.
TAX_CODE	String	Invoice line item tax code. This is populated via the tax determination procedure if a line item is paired.
TAXJURCODE	String	Invoice line tax jurisdiction code. This represents the downstream ERP system ID for the tax office to which tax is payable for this line item. It is used in countries that have tax jurisdictions for their sales tax, and is populated only if the line item is paired.
REF_DOC	String	Reference document number. This is only populated if the line item is paired and the purchase order line item to which it relates was set for goods receipt-based invoice verification.
REF_DOC_YR	String	Year of reference document number. This is only populated if the line item is paired and the purchase order line item to which it relates was set for goods receipt-based invoice verification.
REF_DOC_IT	String	Item in reference document number.
SHEET_NO	String	Service entry sheet number.

Structure Element	Type	Description
SHEET_ITEM	String	Service entry sheet item number.
UNIT_PRICE	Double	Invoice line item unit price. This is only populated for unpaired line items. In all other cases, it will have a value of zero.
DESCRIPTION	String	Invoice line item description. If the line item is paired, the description will be set to the description on the purchase order. If the line is unpaired, this field will contain the raw text description that was extracted from the invoice. For third-party freight invoices, service invoices and MIRA invoices, where no line items were required in the *TAB section*, and line pairing was either not successful for any lines, or was not carried out, the description will be set to **THIRD PARTY FREIGHT**, **SERVICE** or **MIRA** respectively.
MATERIAL_NO	String	Invoice line item material number. If the line item is paired, this will be populated with the material number from the purchase order line item. If the line is not paired, this will be populated with any values extracted from the invoice.
TAX_RATE	String	Invoice line item tax rate. This is the percentage rate of tax applied to the invoice line item. If no percentage tax rate at line item level could be ascertained, then this value will be blank.
LINETYPE	String	Invoice line item type. This is retrieved from the purchase order line item to which an invoice line is paired.
CHARGECODE	String	Invoice line item charge code. This is retrieved from the purchase order line item to which an invoice line is paired.
CHARGECODEID	String	Invoice line item charge code ID. This is retrieved from the purchase order line item to which an invoice line is paired.
MATERIALGROUP	String	Material group. This is retrieved from the purchase order line item to which an invoice line is paired.
DISTILLER_LINE	String	Original Forms Recognition line item number. This is the original line item number in the WebCenter Forms Recognition table field (as viewed in Verifier) that the invoice line was drawn from. This will always be populated for unpaired line items and is a 1-based, not a zero-based index.
PLANT	String	Plant ID. This is retrieved from the purchase order line item to which an invoice line is paired.
COMPANYCODE	String	Company code to which the line item corresponds. This is retrieved from the purchase order when an invoice line is paired.
POTYPE	String	JD Edwards purchase order type. This is populated only if line pairing has been successful for this item.
BUSINESSUNIT	String	PeopleSoft purchasing business unit. This is populated only if line pairing has been successful for this item.

Structure Element	Type	Description
MISCCHARGE	String	An **X** in this element indicates that this is miscellaneous charge line item. The flag will be blank if the system has identified the line as a not being a miscellaneous charge item.

5.3.4.2 POKey Structure

The *POKey* structure contains the elements that comprise the unique key to identify a single purchase order.

The POKey structure is present in the interface to the following user exit:

- UserExitLinePairingPOs

The elements contained within the structure are as follows:

Structure Element	Type	Description
PONUMBER	String	Purchase order number.
COMPANYCODE	String	Company code. This value will only be populated if the company code forms part of the key to identify a unique purchase order (e.g. for JD Edwards implementations).
POEXTENSION	String	Purchase order number extension. This field contains the additional key required to identify a purchase order uniquely. For implementations involving JD Edwards, it contains the purchase order type; for implementations involving PeopleSoft, it contains the purchasing business unit. For any other types of purchase order, this value will be blank.

5.3.4.3 TaxData Structure

The *TaxData* structure holds the tax amounts that are relevant for each tax code. It will be populated if automatic tax determination has been activated and was successful, all lines have been paired and tax is not required to be calculated by the downstream ERP system.

The TaxData structure is present in the interfaces to the following user exits:

- UserExitCustomExport
- UserExitPostLinePairing

The elements contained within the structure are as follows:

Structure Element	Type	Description
TAX_CODE	String	ERP system tax code
TAX_AMOUNT	Double	Tax amount

5.3.4.4 Address Structure

The *Address* structure contains data elements associated with a particular vendor, such as the vendor ID, the vendor name and address details, along with a myriad of additional information. The extent to which the data is populated depends on the extent to which the data is available in the vendor extract, and mapped within the *SRC section* of the project configuration.

The Address structure is present in the interfaces to the following user exits:

- UserExitCustomExport

- UserExitFilterVendorSearch
- UserExitAddressArray

The elements contained within the structure are as follows:

Structure Element	Type	Description
NAME	String	Name of the vendor.
ADDRESS	String	Vendor street address line 1.
ADDRESS2	String	Vendor street address line 2.
ZIP	String	Vendor zip / postal code.
ID	String	Unique vendor ID. This is the unique vendor ID from the point of view of the data extract, where each row must have a unique reference. This will not be the unique vendor ID from the point of view of the ERP system if a site ID is also used.
SITEID	String	Vendor site ID.
TELNO	String	Vendor telephone number.
CITY	String	Vendor city.
STATE	String	Vendor state, county or region. For US addresses, the two-character state code would be expected here. For other countries, the county or region name is expected.
COUNTRY	String	Vendor country. This should be the two-character ISO country code.
POBOX	String	Vendor PO box number.
POBOXZIP	String	Zip / postal code associated with the vendor PO box.
EUMEMBER	Boolean	Boolean indicator denoting whether the vendor belongs to an EU member state country.
VATREGNO	String	Vendor VAT registration number. If the vendor is registered for VAT in more than one country, then multiple VAT registration numbers will be provided in the form of a comma separated list.
TAXID1	String	Vendor tax ID 1.
TAXID2	String	Vendor tax ID 2.
TAXJURCODE	String	ID of the tax office where the vendor is based.
CURR	String	Vendor currency.
INVOICETYPE	String	Vendor invoice type. This will be set to a value that denotes either a PO-supplying vendor or a vendor who submits invoices that legitimately do not reference a purchase order. If this column is being used to determine the invoice type field, the meaning of the values contained in the column should be mapped against the **POValue** and **NPOValue** parameters in the *ITY section* of the project configuration.
PAYMENTMETHODS	String	Comma-separated list of payment method codes appropriate for the vendor.
BANKDETAILS	String	Vendor bank account details. This should be a colon-separated list, in case of multiple bank accounts, in the format:

Structure Element	Type	Description
		BankAccount,SortCode,ERPBankAccountCode A sort code is the US equivalent of a routing number.
UTILITYFLAG	String	Indicator as to whether the vendor is identified as a utility vendor.
PORSUBNO	String	Vendor POR subscriber number (Switzerland only).
EXTERNALID	String	ERP system vendor ID (if a site ID is being used).
ACCOUNTGROUP	String	ERP system vendor account group.
COMPANYCODES	String	Comma-separated list of company codes for which the vendor is valid.
SIRETID	String	Vendor SIRET ID. This is an ID code used in France which uniquely identifies a single vendor at a single address. It is often found on French invoices.

5.3.4.5 Tolerance Structure

The *Tolerance* structure holds a series of limits the system uses when performing mathematical validations on the amounts extracted from the invoice. The tolerance limits are set by the tolerance group assigned to the invoice currency in the project configuration file.

The Tolerance structure is present in the interface to the following user exit:

- UserExitSetTolerance

The elements contained within the structure are as follows:

Structure Element	Type	Description
HEADERTOLERANCE	Double	This is the maximum amount by which the document header amounts (Invoice Total = Tax + Freight + Sum of Line Items or Subtotal) are allowed to deviate from one another before the system marks them as being invalid.
TABLEROWTOLERANCE	Double	This is the maximum amount by which the line item level calculation (Quantity * Unit Price = Total) is allowed to deviate before the system marks a line item as being invalid.
TAXTOLERANCE	Double	If automatic tax determination is activated, then this is the maximum amount by which a system calculated tax amount or tax rate is allowed to deviate from a tax amount extracted from the invoice, or a tax rate contained within the tax table.
NODECIMALPLACES	Boolean	This value will be set to **TRUE** if the invoice currency does not have a sub-unit (e.g. pennies, cents). Common world currencies that do not have sub-units are the Hungarian Forint and the Japanese Yen.

5.3.4.6 Flags Structure

The *Flags* structure contains a range of Boolean indicators for document validation, which can be used to determine which field items are relevant for export.

The Flags structure is present in the interface to the following user exit:

- UserExitCustomExport

The elements contained within the structure are as follows:

Structure Element	Type	Description
MIRA	Boolean	This flag is set to **TRUE** if the invoice is a MIRA and line item extraction is not required for MIRA invoices. The settings in the *TAB section* of the project configuration determine whether

Structure Element	Type	Description
		line items are required under what circumstances.
INVALID	Boolean	This flag is set to **TRUE** if the user has selected the **INVOICE AMOUNTS DO NOT ADD UP** invalid reason in Verifier.
PONOTRELEASED	Boolean	This flag is set to **TRUE** if the PO has not been released and line item extraction is not required for invoices under that circumstance.
NOVENDOR	Boolean	This flag is set to **TRUE** if the user has selected either **VENDOR NOT FOUND** or **MISSING/INVALID VENDOR AND PO** invalid reasons in Verifier.
NOPO	Boolean	This flag is set to **TRUE** if the user has selected either of the **MISSING/ INVALID PO** or the **MISSING/INVALID VENDOR AND PO** invalid reasons in Verifier.
CREDIT	Boolean	This flag is set to **TRUE** if the document type is **CREDIT** and line items are not required for credit memos.
SERVICE	Boolean	This flag is set to **TRUE** if the PO type is **SERVICE** and line items are not required for invoices that relate to service purchase orders.
FI	Boolean	This flag is set to **TRUE** if the invoice type is **NO-PO** and line items are not required for non-PO invoices.
THIRDPARTYFREIGHT	Boolean	This flag is set to **TRUE** if the vendor is third party freight.
NOLINEITEMS	Boolean	This flag is set to **TRUE** if any of the following conditions are met: ▪ Line item extraction is switched off for the project ▪ An invalid reason of **MISSING/INVALID VENDOR AND PO** is set ▪ An invalid reason of **MISSING/INVALID PO** is set and line items are not required under such circumstances ▪ An invalid reason of **VENDOR NOT FOUND** is set and line items are not required under such circumstances ▪ The vendor has been identified as a utility vendor and line items are not required for utility vendors

5.3.4.7 AccountingData Structure

The *AccountingData* structure is used to hold general ledger coding strings relevant to the invoice. It is populated at the time of data export if miscellaneous charges are present on the document, and output is required as a general ledger account entry.

The global project array *GLData*, which is based on this structure, may be populated for either XML or database output in *UserExitPostLinePairing*.

The AccountingData structure is present in the interfaces to the following user exits:

- UserExitCustomExport
- UserExitXMLOutput

The elements contained within the structure are as follows:

Structure Element	Type	Description
INVOICE_DOC_ITEM	Integer	General ledger coding string line item number from 1-n.
PO_NUMBER	String	Purchase order number.
PO_ITEM	String	Purchase order line item number.
GL_ACCOUNT	String	General ledger account number.
COMP_CODE	String	Coding string company code.
DB_CR_IND	String	Debit/credit indicator.

Structure Element	Type	Description
COSTCENTER	String	Cost center.
SERIAL_NO	String	Serial number.
PROFIT_CTR	String	Profit center.
WBS_ELEM	String	Work breakdown structure element.
PROFIT_SEGM_NO	String	Profit segment number.
CO_AREA	String	Controlling area.
CMMT_ITEM	String	Commitment item.
FUNDS_CTR	String	Funds center.
BUS_AREA	String	Business area.
COST_OBJECT	String	Cost object.
FUNC_AREA	String	Functional area.
FUND	String	Fund.
REF_DATE	String	Reference date.
ORDERID	String	Internal order number.
SUB_NUMBER	String	Sub number.
NETWORK	String	Project network.
ACTIVITY	String	Project activity.
RL_EST_KEY	String	Real estate key.
ASSET_NO	String	Asset number.
SD_DOC	String	Sales order document number.
SDOC_ITEM	String	Sales order document item number.
TAX_CODE	String	Tax code.
TAXJURCODE	String	Tax jurisdiction code.
ITEM_AMOUNT	Double	Coding string amount.
QUANTITY	Double	Quantity.
PO_UNIT	Double	Order unit of measure relating to the quantity.
PO_PR_UOM	Double	Order price unit of measure.
PERCENT	Double	Distribution percentage.

5.3.5 Triggering User Exits in Verifier

User exits are triggered when a user is processing a document in the Verifier application. The following table shows the general user exit calls that apply to all fields in Verifier:

Verifier Action	User Exits
User presses the [Enter] key on the Document Type field	UserExitDocumentTypeValidate
User presses the [Enter] key on the Invoice Type field	Always called: ▪ UserExitPOValidateStart ▪ UserExitPOValidateStart2 Potentially called: ▪ UserExitPOValidate (if the PO is validated successfully against a database or ERP)

Verifier Action	User Exits
	▪ UserExitAddressArray (if a vendor has not been loaded into the buffer) ▪ UserExitCheckBankAccount (if the PO is validated successfully against a database)
User presses the [Enter] key on the Invalid Reason field	Always called: ▪ UserExitPOValidateStart ▪ UserExitPOValidateStart2 Potentially called: ▪ UserExitPOValidate (if the PO is validated successfully against a database or ERP) ▪ UserExitAddressArray (if a vendor has not been loaded into the buffer) ▪ UserExitCheckBankAccount (if the PO is validated successfully against a database)
User presses the [Enter] key on the Invoice Number field	UserExitInvoiceNumberValidate
User presses the [Enter] key on the Invoice Date field	UserExitInvoiceDateValidate
User presses the [Enter] key on the Invoice Due Date field	UserExitDueDateValidate
User presses the [Enter] key on the Delivery Date field	UserExitDeliveryDateValidate
User presses the [Enter] key on the Vendor ID field	Always called: ▪ UserExitPOValidateStart ▪ UserExitPOValidateStart2 Potentially called: ▪ UserExitPOValidate (if the PO is validated successfully against a database or ERP) ▪ UserExitAddressArray (if a vendor has not been loaded into the buffer) ▪ UserExitCheckBankAccount (if the invoice type is **NO-PO**', or only if the PO is validated successfully against a database if the invoice type is **PO**)
User presses the [Enter] key on the PO Number or PO Number Extension fields	Always called: ▪ UserExitPOValidateStart ▪ UserExitPOValidateStart2 Potentially called: ▪ UserExitPOValidate (if the PO is validated successfully against a database or ERP) ▪ UserExitCheckBankAccount (if the PO is validated successfully against a database) ▪ UserExitAddressArray (if a vendor has not been loaded into the buffer)
User opens the Vendor Search dialog, or performs a vendor search	Always called: ▪ UserExitFilterVendorSearch Potentially called: ▪ UserExitAddressArray (if a vendor has not been loaded into the buffer)
User clicks on a button on the Verifier form	UserExitDocumentOnAction
User verifies the last invalid field on the Verifier form	Always called: ▪ UserExitDocumentValidate

Verifier Action	User Exits
	Potentially called: ▪ UserExitSetReportingLoginName (if project reporting is activated)
User presses the [Enter] key on the Company Code field	UserExitCompanyCodeValidate
Cursor/field focus is changed or the system moves to a new document	UserExitFocusChanged

5.3.6 Reporting and Custom Base Classes

If the project involves adding a new base class, then the standard reporting trail as written into the AP Project Reporting tables will not be complete without the following steps being performed:

1. Add script into the **Document_PreExtract** and **Document_Validate** events on the custom base class.
2. Create a custom **tmpCLSRES** field in the custom base class.

5.3.6.1 Adding Custom Script to the Document_PreExtract and Document_Validate Events

To insert the script to enabled reporting from the custom base class, follow these steps:

1. Open the project in the Forms Recognition Designer application.
2. Switch to Definition Mode.
3. Right-click the custom base class in the **Classes** tree and select the **Show Script** option from the popup menu. The script window for the custom base class is displayed.
4. Copy-and-paste the following script into the custom base class's script window:

```
Private Sub Document_PreExtract(pWorkdoc As SCBCdrPROJLib.ISCBCdrWorkdoc)

 If InStr(UCase(ScriptModule.ModuleName), cVerifier) Then
  gblVerifierAsServer = True
 Else
  gblVerifierAsServer = False
 End If

 fnGetClassResultsMatrix(pWorkdoc)

 fnReporting(pWorkdoc, "DOCUMENTPREEXTRACT")

End Sub

Private Sub Document_Validate(pWorkdoc As SCBCdrPROJLib.ISCBCdrWorkdoc,
pValid As Boolean)

 gblVerifierAsServer = False

 If UCase(ScriptModule.ModuleName) <> cVerifier Then
  fnReporting(pWorkdoc, "DOCUMENTVALIDATESERVER")
 Else
  fnReporting(pWorkdoc, "DOCUMENTVALIDATEVERIFIER")
 End If

End Sub
```

5.3.6.2 Adding the tmpCLSRES Field

The **tmpCLSRES** field is used to store the full classification results and weightings before they are written to the reporting database. It is very much an internal field and does not require any action beyond its creation. If the field is not created, the full classification results and weightings will not be written into the reporting database, only the final class in which the document was placed. No error will be raised.

Follow these steps to create the **tmpCLSRES** field for the custom base class:

1. In the Designer application's Definition Mode, select the custom base class in the **Classes** tree, then select the **Fields** tab.
2. Right-click the empty space of the **Fields** tab and select the **Insert Field Definition…** option from the popup menu. The **Add Field** dialog is displayed.
3. Enter the new field name as **tmpCLSRES** and click **OK**.

   Note: The field name is case-sensitive and must be entered exactly as shown above.

4. The new field is now displayed on the **Fields** tab. Right-click the field and select **Show Properties** from the popup menu. The field's properties pane is displayed on the right-hand side of the Designer window.
5. Select the **Validation** tab in the field's properties pane.
6. Check the **Always Valid** checkbox.
7. Save the project.

6 Configuring the AP Packaged Project

This section describes the steps required to carry out the basic configuration of the AP Packaged Project.

Upon completion of the steps in this section, the necessary pre-requisites will be in the place to process unseen invoice documents through the system, comprising data extraction and basic output.

Before embarking upon these steps, the WebCenter Forms Recognition implementer should ensure that:

- The software installation of WebCenter Forms Recognition has taken place to the minimum supported version.
- A valid WebCenter Forms Recognition license file has been installed within the <*Installation Folder*>**\Components\Cairo** directory.
- A vendor extract file or data source is available.

6.1 Copy the AP Packaged Project Folder Structure

During a complete installation of WebCenter Forms Recognition, the AP Packaged Project will be installed in a subdirectory of the <*Installation Folder*>**\Projects** folder. Oracle recommends that implementers copy the project subfolder and its entire contents to a working location, rather than configuring and implementing the project from the installation folder.

Once the working location has been created or identified (for example, **C:\WFR Projects**), use Windows Explorer to copy the entire **AP [*version*]** folder from <*Installation Folder*>**\Projects** to that working location.

6.2 Create the Vendor ASSA Pool

The Associative Search Engine is a technology that delivers an unprecedented hit-rate in identifying vendors from documents of unknown structure, by comparing the information on the document to an extract of vendor master data. This comparison is carried out in a *fuzzy* manner, so it is tolerant of:

- OCR problems on the document.
- Differences between the way that vendor details are expressed on the invoice and how they are represented in the vendor master data.
- Vendor names embedded in picture logos.

It is incumbent upon the customer to make the vendor master extract available as either a database table or view (which WebCenter Forms Recognition will connect to via 32-bit ODBC) or in a flat CSV file.

6.2.1 Requirements for the Vendor Extract CSV File

If a flat CSV file is provided, it must be compliant with the following:

- Each row in the file should represent a single vendor at a single address.

- Each row should include, as a minimum, columns that represent the vendor name, the street address, the city, the postcode/zip code and the country.
- The country must be expressed as the appropriate two-character ISO country code.
- Each row in the file must have one column that is a unique identifier for that record and is common only to that row.
- Each row in the file must have an equal number of columns.
- The column separator must be a semi-colon, hence content within a single column will need to be stripped of any semi-colons in advance.
- Each column should be stripped of any double-quotes.
- If the CSV file includes non-Western characters, the file must have an encoding type of **UNICODE**. *ANSI* and *UTF-8* encodings are not supported.

A sample vendor extract that demonstrates the format requirements for a flat CSV file is provided with the AP Packaged Project and is located at:

<*Installation Folder***>\Projects\AP** [*Version*]**\Global\Vendor.csv**

Important Note: If the ERP vendor master data includes intercompany vendors, these must be excluded from the file, as should employees who are set up as vendors. Intercompany vendors are handled in a different way within the solution.

6.2.2 Requirements for the Vendor Extract Database Table or View

If a database table or view is provided, it must be compliant with the following:

- The table or view must be in an ODBC-compliant database
- The appropriate 32-bit ODBC drivers must be installed on the Forms Recognition machine
- An ODBC source must be created on the Forms Recognition machine using the 32-bit ODBC drivers. Oracle recommends that this be created as a *System DSN*.
- Each row in the table or view should represent a single vendor at a single address
- Each row should include, as a minimum, columns that represent the vendor name, the street address, the city, the postcode/zip code and the country
- The country must be expressed as the appropriate two-character ISO country code
- Each row in the table or view must have one column that is a unique identifier for that record and is common only to that row

For customers who use Oracle E-Business Suite as their ERP system, a SQL script is provided with the AP Packaged Project to create the appropriate view in the EBS database. Refer to *Appendix M: Configuring the E-Business Suite Database Views* for more details.

6.2.3 Create, Import and Configure the Vendor ASSA Pool

Refer to *Appendix D: Configuring the Vendor ID Field* for full details of how to create and configure the Vendor ASSA pool appropriately for the requirements of the customer's ERP system.

6.3 Basic Project Configuration Settings

This section describes the basic configuration steps required to get the AP Packaged Project up and running so that invoice documents can be processed.

6.3.1 Configuring Purchase Order Number Formats

If the project implementation involves the extraction of purchase order numbers, then the valid formats must be configured within the *PON section* of the project configuration. If no configuration or an incorrect configuration is made here, then no purchase order numbers will be extracted by the system. More than one purchase order number format can be specified.

Formats are specified using a simple expression, which uses wildcard characters for unknown digits or letters, where a hash (**#**) character represents a number and a commercial at (**@**) character represents a letter. A question mark (**?**) character represents any number, letter or special character. Literal characters can also be included in the simple expression, and the format is not case-sensitive.

As an example, a purchase order number that is always 7 digits in length beginning with **10** would be configured as:

```
PON_VL_01_Format=10#####
```

As a further example, if an additional purchase order number format is used which is 8 characters in length, beginning with two alpha characters and followed by 6 numeric characters, where the number always begins with **14**, the format would be:

```
PON_VL_02_Format=@@14####
```

The *Ignore* parameter that comes alongside each purchase order number format specified, allows the developer to specify special characters that may, or may not, be included within the purchase order number, as shown below:

```
PON_VL_01_Format=10#####
PON_VL_01_Ignore=-
PON_VL_02_Format=@@14####
PON_VL_02_Ignore=-
```

The above configuration would permit a hyphen in any position in a purchase order number matching the defined formats, hence **10-12345**, **1-012345**, **101234-5**, **AB-141234**, **A-B141234** and **AB1-41234** would all be valid matches.

To combat any possible OCR problems, the purchase order number format has a built-in tolerance of 10%; so possible candidates may deviate from the exact expression by that order of magnitude.

When adding a new purchase order number format, it is important to remember to increment the group number of the format group by one.

If the same parameter is found more than once, or the group numbers do run sequentially from 01, the system will fail to load the project configuration correctly and errors will occur during document processing. For example:

```
PON_VL_01_Format=10#####
PON_VL_01_Ignore=-
PON_VL_02_Format=@@14####
PON_VL_01_Ignore=-                ← Duplicate setting
```

```
PON_VL_04_Format=@@24####          ← Non-sequential group numbering
PON_VL_04_Ignore=-                 ← Non-sequential group numbering
```

6.3.2 Configuring Bill-to Name Formats

The Bill-to Name is an optional field within the AP Packaged Project, and a successful extraction will not be required by the system if the following parameter is set as follows:

```
BTO_OP_AllowBlank=YES
```

If the field is required, it is necessary to configure keywords for the bill-to name so that the system is able to identify it, and extract it correctly.

Valid keywords are configured in the *BTO section* of the project configuration in a similar fashion to purchase order number formats described above. A simple expression can be used if required, but, in most cases, merely specifying the first word of a valid bill-to name is sufficient to enable the system to extract the entire field correctly.

For example, the following configuration would be sufficient to extract **Acme**, **ACME**, **Acme Trading Group, Inc.**, **A.C.M.E. CORPORATION**, etc.:

```
BTO_VL_01_Format=Acme
BTO_VL_01_Ignore=.,
```

Note: The formats specified in this section are not case-sensitive.

6.3.3 Configuring Tax Rates

Configuring tax rates is not required, but it is strongly recommended for project implementations that involve processing invoices from countries where VAT or sales tax is set at the national level. For such countries, the *PrimaryTaxRates* and *SecondaryTaxRates* parameters in the *TAX section* of the project configuration should be populated. Doing so will increase the success of the tax amount extraction by a noticeable margin. This will also enhance the extraction of tax percentage rates specified on the invoice at line item level.

This configuration does not apply to countries that use tax jurisdictions, where the rate of sales tax can vary depending on the location in the country where the goods or services are consumed. Examples of such countries are the US, Canada and Brazil. For projects only involving these countries, the parameters above should always be left blank.

A primary tax rate refers to the standard rate of tax applied to purchases in those countries where the tax rates are set at a national level. Some examples of this include 17.5% or 20% in the UK, 19% in The Netherlands and Germany or 19.6% in France. A secondary tax rate refers to the reduced rate of VAT applied to certain goods or services in those countries, for example, 5% in the UK, 6% in The Netherlands, 7% in Germany and 5.5% or 2.1% in France.

An example configuration covering the UK, Germany, France and The Netherlands, based on the rates listed above would be:

```
TAX_VL_PrimaryRates=17.5,20,19,19.6
TAX_VL_SecondaryRates=5,6,7,5.5,2.1
```

6.4 Configuring a Standard Output File

The output of the AP Packaged Project is governed by the settings in the *EXP section* of the project configuration. Within this section, there are configuration options to output data in a variety of configurable flat files, or to write the results to a set of database tables.

For the purposes of a basic installation, the system can be instructed to output a standard results file via the following parameter:

```
EXP_OP_OutputStandardResultsFile=YES
```

Additional configuration can also be applied to determine the format of dates within that file. The following parameters are relevant for this:

```
EXP_VL_OutputDateFormat=
EXP_VL_OutputDateSeparator=
```

The valid settings for the *OutputDateFormat* parameter are **YYYYMMDD**, **MMDDYYYY** or **DDMMYYYY**. If this parameter is left blank, or an invalid format is specified, the system will default to an output format of **DDMMYYYY**.

It is also possible to configure a separator character. Single or double quotes, and the greater than or less than symbols are not permitted as separators, and, if specified, will not be visible in the output.

For example, an invoice date is stated on the document as **2nd November 2008**. The following configuration will define the output date as **11/02/2008**:

```
EXP_VL_OutputDateFormat=MMDDYYYY
EXP_VL_OutputDateSeparator=/
```

6.5 Setting up the Runtime Server Instance

The WebCenter Forms Recognition Runtime Server (RTS) is the software component that processes documents at runtime. It runs as a Windows service and is configured and controlled using a snap-in to the Microsoft Management Console.

The management console is opened from the Windows Start menu as follows:

> *All Programs* ➤ *Oracle* ➤ *WebCenter Forms Recognition* ➤ *Runtime Service* ➤ *Management Console*

If the WebCenter Forms Recognition Runtime Service Manager service has not been started, then the following will start the service and open the management console:

> *All Programs* ➤ *Oracle* ➤ *WebCenter Forms Recognition* ➤ *Runtime Service* ➤ *Start Runtime Service*

The first step is to create an RTS group. An RTS group represents a WebCenter Forms Recognition environment, and consists of one or more physical or virtual machines and the individual RTS instances that run on those machines.

To create an RTS group:

1. Ensure that the WebCenter Forms Recognition Runtime Service Manager service has been started and the management console is open.
2. In the management console, right-click on the **Runtime Server Administration** node and select **New RTS Group…** from the popup menu. The *New* Group dialog is displayed.

3. Enter the desired name of the RTS group in the *New Group* field, for example, **My RTS Group**.

Once the RTS group has been created, it is possible to add individual machines to it. WebCenter Forms Recognition employs a scalable server architecture, which permits the processor power of multiple machines to be combined to process the required number of documents. This functions as a master/slave server relationship where it is possible to add as many slave servers as required to the master server, which controls the environment licensing.

For the purposes of a basic installation, it is assumed that the Forms Recognition environment will consist of a single master server machine. To add this machine to the newly created RTS group:

1. In the management console, right-click on the RTS group and select the **New Machine…** option from the popup menu. After a short delay (while the RTS management service searches the domain for other machines running Forms Recognition services) the *Group Management* dialog is displayed.
2. In the *Group Management* dialog, select the domain of the machine to be added from the drop down list (or leave as the default if the local machine is to be added) and enter the technical name of the computer in the *Available WebCenter Forms Recognition Machines* field.
3. Click the **OK** button. The *Group Management* dialog will close, and the specified machine will appear as a child node to the RTS group.

The final step is to create and configure an RTS instance. An RTS instance is a process (viewable in Task Manager as **dsthost.exe**) that carries out one or more steps of the WebCenter Forms Recognition workflow, namely Import, OCR, Classification, Extraction, Export and Clean-Up.

A single machine may have more than one RTS instance created for it, with each instance carrying out different steps in the workflow. It is recommended that the number of instances carrying out OCR, classification and extraction, which are highly resource-intensive, do not exceed the number of available processors on the machine. For a quad-CPU machine, a typical production configuration on the master server would be one instance performing import, export and cleanup, with the other three performing OCR, classification and extraction.

In the following example, only one RTS instance will be configured and will perform the Import, OCR, Classification, Extraction and Export workflow steps:

1. In the management console, right-click on the machine node and select **New ▸ RTS Instance…** from the popup menu. The *New RTS Instance* dialog is displayed.
2. Enter the desired name for the RTS instance and click the **OK** button. The *New RTS Instance* dialog is closed and the RTS instance will be added as a child node to the machine.

Note: It is common practice to make reference to workflow steps that the instance carries out in the instance name.

---

3. To set the properties of the instance, right click on the instance name and select **Properties…** from the popup menu. The Properties dialog is displayed.
4. Complete the fields on the **General** tab of the properties as follows, paying due respect to the file paths and directories that were established in *section 6.1* above. The critical file paths to be populated are the path to the project file and the path to the export directory.

In a production environment, file paths should always be expressed as a UNC path and the project directory should have a sufficient level of share permissions.

The **Wait** property denotes the time interval between scans of the batch and import directories.

The **Enable Batch Integrity** option should only be selected on one RTS instance per batch directory.

5. On the **Workflow** tab, select the workflow steps that the instance is to perform. For this example, ensure that only the **Import**, **OCR**, **Classification**, **Extraction** and **Export** buttons are depressed.
6. On the **Import** tab, enter the path to the import directory, and ensure that the two radio button groups are set to **1 batch per subdirectory, 1 folder per batch** and **Always import documents**. If the file types to be imported are TIFF files, then the *Document Type* section can be left as the default values. For other document types, the **Automatic** checkbox should be checked.

   There is also an opportunity here to limit the batch size. Oracle recommends that this should be set to 10 documents.
7. On the **OCR+Export+Clean Up** tab, ensure that only the **Trigger script based export** option is checked in the *Export* section.
8. Click the **OK** button. The configuration of the RTS instance is now complete, and the *Properties* dialog is closed.

6.6 Setting up the Verifier Application

The WebCenter Forms Recognition Verifier module is a client application that can either be installed directly on a user's local machine, or it can be installed on a central server and distributed to the users via Citrix.

As documents are processed through the system, if one or more fields on one or more documents in a batch are found to be invalid for whatever reason, the batch will be stopped at state 550 for a user to review in Verifier.

Verifier is opened from the Windows Start menu as follows:

> *All Programs* ➤ *Oracle* ➤ *WebCenter Forms Recognition* ➤ *WebCenter Forms Recognition Verifier*

The system will prompt for a project login. Once a username and password has been entered, the Verifier initial screen will appear.

On the Verifier initial screen, the user is provided with a list of all batches that require further examination. The status of a batch represents its position in the Forms Recognition workflow, and a batch will always inherit the status of the document with the lowest status within it. For example, if a batch contains 10 documents, and 9 of them have a status of 700, but 1 has a status of 550; the overall batch status will be 550.

A list of standard batch states, along with their corresponding meanings, is provided in the table below:

Batch State	Meaning
0	Corrupt batch. Unable to process
100	Documents have been imported successfully and the batch is awaiting OCR.

Batch State	Meaning
150	The OCR workflow step failed.
200	The batch has been successfully OCR'd and is awaiting classification.
250	One or more documents in the batch failed classification. Manual intervention using Verifier is required.
300	The batch has been successfully classified and is awaiting field extraction.
550	One or more mandatory fields on one or more documents failed extraction or validation. Manual intervention using Verifier is required.
601 - 699	The batch is at a user-designated exception state.
700	All fields on all documents in the batch have been successfully extracted and validated, and the batch is awaiting export.
750	One or more documents in the batch failed the export step. Manual intervention is required from a system administrator.
800	The batch was successfully exported and can be cleaned up.

Upon entering a batch, the system will navigate the user to the first problem document within the batch, and the cursor focus will be set to the first invalid field. An error message visible in the application status bar will explain why the field could not be validated.

If the document is at a state of 750, which denotes a failed export, then the reason for the export failure is displayed as an error message against the Invoice Number field.

All error messages, except those that are generated by external systems and relayed back into the Verifier application, are configurable in the *ERR section* of the project configuration.

To configure the Verifier application:

1. Select the **Settings…** option from the **Options** menu. The *Properties* dialog is displayed.
2. On the **General** tab, enter the file paths to the project file and the batch directory, or select the appropriate batch job from the dropdown list.

   A useful feature on this tab is the *Fields edit mode* radio button group. This defaults to **Insert**, which means that when a user clicks into a field, the application will position the cursor in the existing text. If the **Overwrite** option is selected, the entire contents of the field will be highlighted allowing the user to key fresh data into the field without having to remove the existing field content.
3. On the **Workflow** tab, relevant batch states for Verifier processing are specified. To add, remove or change an input state, simply right click on the *Input State* list and select the appropriate option from the context menu.

   In a basic installation, the extraction verification input states are typically set to **550** and **750**.
4. Click the **OK** button. The *Properties* dialog is closed.
5. Select the **Save** option from the **File** menu to save the settings. The configuration of the Verifier application for a basic installation is now complete.

6.7 Processing Documents Through the System

When the configuration of the system is complete, it is possible to process documents through the system. To do this:

1. Place the appropriate documents into the import directory specified in the RTS properties configured in *Section 6.5: Setting up the Runtime Server Instance*.

Note: If TIFF files are to be processed, these should be, at an optimum, 300 dpi with a corresponding pixilation count. The TIFFs should also be black and white using Group 4 compression. If the document is a multi-page invoice, the file should be a multi-page TIFF file with the pages in the correct order.

---

2. Open the Runtime Server management console, select the configured RTS instance, and click the green *start* button in the toolbar. The system will now import the documents placed in the import directory, and will commence the OCR, classification and extraction operations.
3. Once complete, if there was a problem with one or more documents, the batch will be available in the Verifier application.
4. Open the Verifier application and login to the project.
5. To get an overview of how the system has performed, highlight the batch, and select **Show Selected Batch** from the **View** menu. The batch overview screen is displayed.

   This status screen provides an overview of all of the individual document states in the batch.
6. To enter the batch from this screen, select **Verify Selected Batch** from the **View** menu.

   Any documents that require manual verification will be displayed in Verifier with the invalid fields displayed in red. The corresponding error message for the selected field is displayed in the lower status bar.
7. Correct any validation failures and move onto the next invalid document in the batch, until all documents in the batch are complete.
8. When all documents in the batch have been validated, the system will prompt the user to release the batch, which will provide the user with three options: **Yes**, **No**, and **Details**.

   If the user selects **Yes**, the batch state will be set to **700**, which means that it is ready for export.

   If the user selects **No**, the batch is not released and will remain in the current verification state.

   Clicking the **Details** button provides a radio button group that controls whether the system will move on to the next invalid batch, or whether the system will return to the batch overview screen.
9. Once ready for export, the Runtime Server instance will pick the batch up again, and will carry out the export operation.
10. Once complete, navigate to the export directory specified in the RTS properties configured in *Section 6.5: Setting up the Runtime Server Instance*. The directory will now contain a number standard output files, one for each of the processed documents:
11. Document processing is now complete.

7 Configuring Data Export

7.1 Introduction

This section covers the steps required to configure the various export options available within the AP Packaged Project.

The types of export covered by this section are:

1. Outputting an additional TIFF image
2. Outputting a PDF file
3. Writing data to database tables
4. Writing data to an XML file
5. Writing data to a CSV file
6. Setting up a custom export

The standard solution exports only apply for documents classified to the **Invoices** class, or to one of its child classes. For custom base classes, the data export will need to be coded programmatically within **UserExitCustomExport** on the *UserExits* script class level.

7.2 Outputting an Additional TIFF Image

The standard Runtime Server (RTS) instance settings contain an option for outputting the TIFF image into the export directory, and this setting should be used over custom settings in the project configuration file where appropriate.

The custom settings, however, provide additional options for the TIFF image, and should be used in cases where one or more of the following applies:

1. A second TIFF file is required during document export.
2. The naming of the TIFF file should be set to the document *URN*, rather than the original image filename.
3. The resolution of the TIFF image needs to be changed from that of the original image.
4. The compression of the TIFF image needs to be changed from that of the original image.

To output an additional TIFF image, the following parameter In the *EXP section* of the project configuration should be set to **YES**:

```
EXP_OP_OutputTiffFile=YES
```

To set the name of the TIFF file to the document *URN* as configured in the *IMP section* of the project configuration, the following parameter should be set to **URN**:

```
EXP_VL_TiffName=URN
```

If the above parameter is left blank, or set to anything other than **URN**, the TIFF filename will retain the same name as the original imported document.

The following parameter controls the resolution in *dots per inch* that will apply to the TIFF file:

```
EXP_VL_TiffDPI=300
```

If a blank or invalid entry is set against the above parameter, the system will default to 300dpi.

To change the image compression type, the following options may be set against the *EXP_VLTiffFormat* parameter:

Parameter Value	Compression Type Description
G4FAX	Group 4 compression
G3FAX	Group 3 compression
LZWFAX	LXW compression
HUFFAX	HUF compression

For example, to configure the output TIFF to use Group 3 compression, the parameter should be set as follows:

```
EXP_VL_TiffFormat=G3FAX
```

If a blank or invalid entry is set against the above parameter, the system will default to standard Group 4 compression.

The additional TIFF file will always be written to the directory specified as the export directory on the RTS instance settings for the instance that is carrying out the document export. If no export directory has been specified, the default export directory set against the following parameter will be used:

```
EXP_VL_DefaultExportPath=<default export path>
```

If the above parameter is also blank, document export will fail and the batch will go to status **750**.

7.3 Outputting a PDF File

To configure the system to output a searchable PDF document for each document processed, the following parameter should be set to **YES**:

```
EXP_VL_OutputPDF=YES
```

To set the name of the PDF file to the document *URN* as configured in the *IMP section* of the project configuration, the following parameter should be set to **URN**:

```
EXP_VL_PDFName=URN
```

The PDF file will always be written to the directory specified as the export directory on the RTS instance settings for the instance that is carrying out the document export. If no export directory has been specified, the default export directory set against the following parameter will be used:

```
EXP_VL_DefaultExportPath=<default export path>
```

If the above parameter is also blank, document export will fail and the batch will go to status **750**.

7.4 Writing Data to Database Tables

The following sections describe the process of setting up a database export, and how the export of the header data can be customized to add additional fields.

7.4.1 Configuring Database Export

To activate output to a database, the following parameter must be set to **YES**:

```
EXP_OP_ExportToDB=YES
```

In addition, a valid SQL connection group containing a valid database connection string in the *SQL section* of the project configuration must be specified against the following parameter:

```
EXP_VL_SQLConnectionGroup=01
```

The database export can write into three different types of table:

1. An invoice header data table.
2. An invoice line item data table.
3. An invoice general ledger account coding table.

7.4.1.1 Invoice Header Table

At a minimum, output to an invoice header database table must be configured. Invoice header information includes data items that apply to the document as a whole, such as the invoice number, the invoice date and the vendor.

The structure of the destination invoice header database table should be such that each row has a single column that represents that unique key for the database record. By default, the system will populate this column with the name of the image filename minus the file extension, but the document *URN* as mapped in the *IMP section* of the project configuration can also be used via the following parameter:

```
EXP_VL_DBKey=URN
```

To configure that name of the destination invoice header table, and to set the name of the column in that table which represents the unique record identifier, the following parameters are used:

```
EXP_VL_DBHeaderTable=MYHEADERTABLE
EXP_VL_DBHeaderKey=MYKEY
```

In the example above, the system is configured to write invoice header information into a table with the technical name of **MYHEADERTABLE**, where **MYKEY** is the technical name of the column designated as the unique record identifier in the table.

Invoice header information can be written into the table as a new *insert* or as an *update* to an existing record that was, for example, initialized during the scanning procedure. This is controlled by the following parameter, which should be set to either **INSERT** or **UPDATE**:

```
EXP_VL_DBHeaderOperation=INSERT
```

Once the basic configuration above has been completed, it is now possible to specify which fields should be outputted and their destination columns.

In the project configuration, there are over 40 header fields available to be mapped. Each of these has a corresponding parameter with the prefix **EXP_VL_DBHC**. To indicate that the field should be outputted, and to specify the technical name of the destination database table column, the appropriate parameter should be configured as follows:

```
EXP_VL_DBHCInvoiceNumber=INVOICENO
```

In the example above, the extracted invoice number will be written into a column in the database header table with a technical name of **INVOICENO**.

If the parameter is left blank, then the invoice number will not be written into the database table, for example:

```
EXP_VL_DBHCInvoiceNumber=
```

This process should be repeated for all header fields for which database output is required, ensuring that the parameters are left blank for any fields that are not relevant for database output.

7.4.1.2 Invoice Line Items Table

The output of invoice line items is optional. The system expects the destination line item table to have a composite key consisting of two fields:

- The document identifier, which will be populated with a value identical to that passed into the header table unique key field, which must have the same technical column name.
- A line item number index column, which the system will set from **1**-*n*, where *n* is the number of line items.

To configure the output of line items, the following parameters must be configured:

```
EXP_VL_DBLineItemsTable=MYLINEITEMS
EXP_VL_DBLineItemsKey=LINEID
EXP_VL_DBHeaderKey=MYKEY
```

In the above example, the system will write the line item data into the database table called **MYLINEITEMS**, where the document identifier is the **MYKEY** column and the line item number field is called **LINEID**. The database table columns **MYKEY** and **LINEID** come together to form the unique row identifier for an entry in the line data table.

If no table is specified, no line items will be written to the database. If a table is specified, but no line items key field is nominated, then the export will fail and the batch will go to a state of 750.

Additionally, the system will not write out line item data if any of the following conditions are met for any given document:

1. Line items are not subject to validation as set in the *TAB section* of the project configuration.
2. The document is a credit memo, and line items are not required for credit notes.
3. The document is a non-PO, and line items are not required for non-PO invoices.
4. The document is PO-related, but the PO has not been released, and line items are not required for documents with an unreleased PO.
5. The user selected an **INVOICE AMOUNTS DO NOT ADD UP** invalid reason in Verifier.

6. The user selected either a **VENDOR NOT FOUND**, **MISSING/INVALID PO** or a **MISSING/INVALID VENDOR AND PO** invalid reason in Verifier, and line items are not required for that particular invalid reason.
7. The document is from a utility vendor, and line items are not required for utility vendors.

For MIRA, third-party freight and service PO invoices where line items were not required and line pairing was either unsuccessful or deactivated, a single line item will be written into the database line items table with a quantity of **1**, and a net price and line item total set to the net invoice value. The description is set to **MIRA**, **SERVICE** or **THIRD PARTY FREIGHT** as appropriate.

In the project configuration, there are over 25 line item fields available to be mapped. Each of these has a corresponding parameter with the prefix **EXP_VL_DBTable**. To indicate that the field should be outputted, and to specify the technical name of the destination database table column, the appropriate parameter should be configured as follows:

```
EXP_VL_DBTableUnitPrice=UNIT_PRICE
```

In the example above, the extracted line item unit price will be written into a column in the database line items table with a technical name of **UNIT_PRICE**.

If the parameter is left blank, then the unit price will not be written into the database table, for example:

```
EXP_VL_DBTableUnitPrice=
```

This process should be repeated for all line item fields for which database output is required, ensuring that the parameters are left blank for any fields that are not relevant for database output.

When writing the line item information to the database, the system will first remove any existing line item records that are present for the document in question. If the line item export to the database is not successful, the entire document record will be rolled back, which includes the header level entry and document export will fail, sending the batch to a status of 750.

7.4.1.3 General Ledger Coding Items Table

The output of general ledger line items is optional. The system expects the destination GL table to have a composite key consisting of two fields:

- The document identifier, which will be populated with a value identical to that passed into the header table unique key field, which must have the same technical column name.
- A GL line item number index column, which the system will set from 1-*n*, where *n* is the number of general ledger line items.

To configure the output of GL items, the following parameters must be configured:

```
EXP_VL_DBGLItemsTable=MYGLACCOUNTS
EXP_VL_DBGLItemsKey=GLLINEID
EXP_VL_DBHeaderKey=MYKEY
```

In the above example, the system will write the GL item data into the database table **MYGLACCOUNTS**, where the document identifier is the **MYKEY** column and the line item number field is **GLLINEID**. The database table columns **MYKEY** and **GLLINEID** come together to form the unique row identifier for an entry in the GL line item table.

If no table is specified, no GL items will be written out. If a table is specified, but no GL items key field is nominated, then the export will fail and the batch will go to a state of 750.

In the project configuration, there are over 25 GL item fields available to be mapped. Each of these has a corresponding parameter with the prefix **EXP_VL_DBGLTable**. To indicate that the field should be outputted, and to specify the technical name of the destination database table column, the appropriate parameter should be configured as follows:

```
EXP_VL_DBGLTableCostCenter=COST_CENTER
```

In the example above, the GL line cost center will be written into a column in the database GL item table with a technical name of **COST_CENTER**.

If the parameter is left blank, then the cost center will not be written into the database table, for example:

```
EXP_VL_DBGLTableCostCenter=
```

This process should be repeated for all GL item fields for which database output is required, ensuring that the parameters are left blank for any fields that are not relevant for database output.

When writing the line item information to the database, the system will first remove any existing GL item records that are present for the document in question. If the GL item export to the database is not successful, the entire document record will be rolled back, which includes the header level entry and document export will fail, sending the batch to a status of 750.

7.4.2 Adding Additional Fields to the Database Header Export

It is possible to add custom fields and values into the database header export.

Assume a new field has been created on the *Invoices* class level called **InvoiceCode**, and it is required to add this new field into the database header export. The process consists of the two steps provided in the following sections.

7.4.2.1 Adding a New Project Configuration Parameter

A new parameter relating to the new header field must be added to the project configuration. This parameter, in common with the delivered database header export configuration parameters, should be prefixed with **EXP_VL_DBHC**. The exact naming of the parameter does not necessarily have to equate to the technical name of the field it relates to, but it should be something meaningful. The name of the parameter must also be unique within the project configuration.

In this example, a new parameter is created for the newly added *InvoiceCode* field, which is to be written to column **INVOICE_CODE** in the invoice header database table. The parameter should be entered into the project configuration as follows:

```
EXP_VL_DBHCInvCode=INVOICE_CODE
```

7.4.2.2 Adding Script into the Database Export User Exit

Script must also be added to the database header export user exit to ensure that the custom field value is written to the invoice header table during export. To do this:

1. Open the project file with the Designer application.
2. Switch to Definition Mode.

3. Right-click the **UserExits** class in the *Classes* tree and select **Show Script** from the context menu.
4. In the script window, navigate to the user exit sub-routine named U**serExitDBHeader Export**.
5. Add the following line of script into the *UserExitDBHeaderExport* subroutine:

   ```
   fnWriteDBHeaderField("InvCode", pWorkdoc.Fields("InvoiceCode").Text,
   strINIfileKeyName, strFieldValue)
   ```

   The *fnWriteDBHeaderField* function has been specially provided for this purpose, to condense the operation into a single command.
6. Close the script window and save the project.

Note: If the database field to be added is a date field, and that date field is in the Verifier output format configured in the *DAT section* of the project configuration, then the function **fnWriteDBHeaderDateField** should be used instead. This function has an identical interface, but will convert the date into the export output format configured in the *EXP section* of the project configuration.

7.5 Writing Data to an XML File

The following sections describe the process of configuring XML export, and how the XML file can be set up to include custom fields.

7.5.1 Configuring the XML File

The standard XML output file is divided into four separate sections:

1. The document section.
2. The invoice header section.
3. The invoice line items section.
4. The GL coding item section.

The document section includes global document fields divorced from the type of business document it represents, for example the scan date, the priority flag, the URN etc.

The invoice header section includes fields associated with the invoice header such as the invoice number, the invoice date, the vendor ID etc.

In the invoice line items section, the invoice line item information is written line-by-line, and includes information such as the line item quantity, the unit price, the line item total and material number.

The general ledger coding section contains all of the GL account entries associated with the invoices, which includes information such as the general ledger account number, the cost center, the tax code and the amount.

Each of these four sections has a specific tag, as does each field within those sections. All tags are configurable within the project configuration, as is the decision point concerning which fields are written into the file and which fields are not.

The basic structure of the file is as follows:

```
<invoiceImport>                    ← Configured via EXP_VL_XMLFileHeader

    [Document level fields]

    <header>                       ← Configured via EXP_VL_XMLInvoiceHeader
        [Invoice header fields]
    </header>

    <lineItems>                    ← Configured via EXP_VL_XMLLineItemsHeader
        [Invoice line item fields – one set of entries per line item]
    </lineItems>

    <GLLines>                      ← Configured via EXP_VL_XMLGLLinesHeader
        [GL item fields – one set of entries per GL line]
    </GLLines>

</invoiceImport>
```

To activate output of an XML file, the following parameter must be set to **YES** in the project configuration:

```
EXP_OP_OutputXMLFile=YES
```

To set the name of the XML file to the document URN as configured in the *IMP section* of the project configuration the following parameter should be set to **URN**.

```
EXP_VL_XMLFileName=URN
```

It is also possible to set the file extension of XML output via the following parameter:

```
EXP_VL_XMLFileType=
```

If left blank, the file extension will default to **.XML**.

It is possible to add an encoding header to the XML file via the following example:

```
EXP_VL_XMLEncodingHeader=<?xml version="1.0" encoding="UTF-16"?>
```

In the above example, the encoding header is set to support UTF-16, which will permit non-Western characters to be written into the XML file. The XML file will not be generated successfully if the output data includes non-Western characters and the encoding is set to use UTF-8.

The XML file will always be written to the directory specified as the export directory in the Runtime Server instance settings for the instance that is carrying out the document export. If no export directory has been specified, the default export directory set against the following parameter will be used:

```
EXP_VL_DefaultExportPath=<default export path>
```

If this is also blank, document export will fail and the batch will go to status 750.

Once the XML output has been activated, and the section tags configured, it is now possible to elect which fields should be written into the file, and define how they are tagged.

In the project configuration, all fields that are available for output across all of the four sections have a corresponding parameter:

- Fields in the document section have a prefix of **EXP_VL_XML**.

- Fields in the invoice header section have a prefix of **EXP_VL_XMLHC**.
- Fields in the line items section have a prefix of **EXP_VL_XMLTable**.
- Fields in the GL items section have a prefix of **EXP_VL_XMLGLTable**.

To add a field into the XML file, simply add the desired tag to the field parameter.

For example, if the invoice number is to be written into the XML file, and this should have a tag of **invNo**, then the invoice number parameter should be set as follows:

```
EXP_VL_XMLHCInvoiceNumber=invNo
```

If the content of the field is **12345**, then this will have the effect of writing the following into the invoice header section of the XML file:

```
<invNo>12345</invNo>
```

If a field is not to be written into the XML file, then its corresponding parameter should be left blank, for example:

```
EXP_VL_XMLHCInvoiceNumber=
```

This process should be repeated for all fields in the XML section.

Section tags for line items and GL line items will only be written into the XML file if there is active content to be placed within those tags.

The system will not write out line item data into the XML file if any of the following conditions are met by any given document:

8. Line items are not subject to validation as set in the *TAB section* of the project configuration.
9. The document is a credit memo, and line items are not required for credit notes.
10. The document is a non-PO, and line items are not required for non-PO invoices.
11. The document is PO-related, but the PO has not been released, and line items are not required for documents with an unreleased PO.
12. The user selected an **INVOICE AMOUNTS DO NOT ADD UP** invalid reason in Verifier.
13. The user selected either a **VENDOR NOT FOUND**, **MISSING/INVALID PO** or a **MISSING/INVALID VENDOR AND PO** invalid reason in Verifier, and line items are not required for that particular invalid reason.
14. The document is from a utility vendor, and line items are not required for utility vendors.

For MIRA, third-party freight and service PO invoices where line items were not required and line pairing was either unsuccessful or deactivated, a single line item will be written into the XML file with a quantity of **1**, and a net price and line item total set to the net invoice value. The description is set to **MIRA**, **SERVICE** or **THIRD PARTY FREIGHT** as appropriate.

7.5.2 Adding a Field into the XML File

The system provides a simple method to insert custom fields into any section of the XML file.

In this example, a new field has been created on the *Invoices* class level called **InvoiceCode**. It is required to add this new field into the invoice header section of the XML file.

The process consists of the two steps provided in the following sections.

7.5.2.1 Adding a New Project Configuration Parameter

The first step is to add a new parameter into the project configuration. This parameter, in common with the delivered XML invoice header configuration parameters, should be prefixed with **EXP_VL_XMLHC**. The exact naming of the parameter does not necessarily have to equate to the technical name of the field it relates to, but it should be something meaningful. The name of the parameter must also be unique within the project configuration.

In this example, a new parameter is created for the newly added **InvoiceCode** field, which is to be written into the XML file with a tag of **invCode**. The parameter should be entered into the project configuration as follows:

```
EXP_VL_XMLHCInvoiceCode=invCode
```

7.5.2.2 Adding Script into the XML Export User Exit

Follow the steps below to add script to the XML output user exit:

1. Open the project file in the Designer application.
2. Swicth to Definition mode.
3. Right-click the **UserExits** class and select **Show Script** from the context menu.
4. In the script window, navigate to the user exit subroutine named **UserExitXMLOutput**.
5. Add the following line of script into the subroutine. This line should go in the **cXMLInvHeader** clause of the subroutine's *Select Case* structure:

   ```
   fnWriteXMLField("HCInvoiceCode", pWorkdoc.Fields("InvoiceCode").Text)
   ```

   Note: The *fnWriteXMLField* function has been specially provided for this purpose to condense the operation into a single command.

6. Close the script window and save the project.

The new field has now been added to the XML output in the invoice header section, and, if the field content is **12345**, will be written to the XML output as follows:

```
<InvCode>12345</InvCode>
```

When specifying the project configuration parameter, only the **XML** prefix can be omitted.

For example, if the parameter is **EXP_VL_XMLMyField**, then the parameter passed to the *fnWriteXMLField* function should be **MyField**. If the parameter is **EXP_VL_XMLHCMyField**, then the parameter passed should be **HCMyField**. If the parameter is **EXP_VL_XMLTableMyField**, then the parameter passed should be **TableMyField**.

If the field to be added is a date field, and that date field is in the Verifier output format configured in the *DAT section* of the project configuration, then function *fnWriteXMLDateField* should be used instead. This function has an identical interface, but will convert the date into the export output format configured in the *EXP section* of the project configuration.

7.6 Writing Data to a CSV File

The following sections describe how output to a CSV file can be configured. The structure of the CSV file is entirely configurable up to five lines per document at header level, and one additional line for each line item. It is also possible to configure CSV file output at the document or batch level, and to create multiple files depending on whether the invoice type is **PO** or **NO-PO**.

It is possible to output up to 99 different CSV files per document processed.

7.6.1 Configuring the CSV File

Configuration of CSV file output is performed in the *CSV section* of the project configuration.

To activate CSV file output, the following parameter should be set to **YES**:

```
CSV_OP_OutputCSVFile=YES
```

This controls CSV file output globally as a convenient switch to activate and deactivate all output files configured.

The *CSV section* is divided into numbered groups (**01** to **99**), with each group defining the output for a single CSV file. Should an additional CSV file be required, adding a new group into the project configuration will achieve this.

To activate CSV file output for group **01**, the following parameter should be set to **YES**:

```
CSV_OP_01_OutputFile=YES
```

The output file will be written to the export directory configured against the CSV file group *Filepath* parameter. If no file path is specified, the system will use the export file path set against the RTS instance carrying out the export step in the project workflow. If no export directory is specified in the RTS instance properties, the default export file path set in the *EXP section* of the project configuration will be used. If this is also blank, or any file path is invalid, export of the CSV file will fail and the document will go to state 750.

The naming of the file depends upon whether output is required on a per-document or a per-batch basis. This is controlled via the *CombinedFilePerBatch* parameter, which should be set to **YES** if one file per batch is required, or **NO** if one CSV file per document is required.

Note: In order for the combined file per batch option to take effect, the Document Grouping option for import in the RTS instance settings must be set to **1 folder per batch** or **1 batch per subdirectory, 1 folder per batch**.

For an individual CSV file per document, the filename will follow the naming convention: <*FilePrefix*><*filename or URN*>.<*FileType*>

This naming convention observes the following project configurations for the group:

- CSV_VL_nn_FilePrefix
- CSV_VL_nn_FileName
- CSV_VL_nn_FileType

For a CSV file per batch, the filename will follow the naming convention: <*FilePrefix*><*8-digit batch id*>.<*FileType*>

This naming convention observes the following project configurations for the group:

- CSV_VL_nn_FilePrefix
- CSV_VL_nn_FileType

It is also possible to specify a date format and separator for each individual CSV file outputted. Date formats can be **YYYYMMDD**, **MMDDYYYY** or **DDMMYYYY**. If no date format is specified then the output date format configured in the *EXP section* of the project configuration will be used. If this is also blank, the system will output all dates as **DDMMYYYY**.

For example, the CSV group has the following settings:

```
CSV_VL_01_DateFormat=YYYYMMDD
CSV_VL_01_DateSeparator=-
```

If the date were 2nd November 2009, it would be outputted as **2009-11-02**.

The standard amount fields are always outputted using a period/full-stop as the decimal separator.

CSV file output can be restricted depending on the contents of the project's *Invoice Type* field, which denotes whether the document is purchase order-related or not. The group *InvoiceType* parameter is provided for this, and can be set to **PO** or **NPO**.

For example, if the CSV file is only to be created for purchase order-related invoices, the parameter should be set as follows:

```
CSV_VL_01_InvoiceType=PO
```

If the parameter is left blank, then the file will be outputted for both purchase order and non-purchase order related invoices. If any other value is set, no CSV file will be outputted.

To output an additional copy of the original image to the destination directory, the following parameter should be set to **YES**:

```
CSV_OP_01_OutputImage=YES
```

The structure of the CSV file is determined by the configuration set against the *FormatLineN* parameters, where *N* denotes the line number in the CSV file. Each document entry at header level in the CSV can span up to five lines. Any lines with no corresponding configuration denote the end of the header record entry.

Values set against the *FormatLineN* parameters are essentially freetext, where literals are used to specify where extracted values from the document should be inserted into that particular line. See *Section 7.6.4: Available Literals for Configuring the CSV Format* for a full list of the literals that are supported by the delivered AP Packaged Project.

If a CSV file that uses a simple separator is required, populating the CSV file group *Separator* parameter will remove any instances of that separator from the fields that are to be written into the file. The separator must still be included in the line format string. It is not possible to specify a period/full-stop, a forward slash or a backslash as a separator.

If line item detail is also required in the CSV file, this is populated against the CSV file group *LineItem* parameter. One line will be written into the CSV file for each line item. The formulation of the line item detail follows the same rules as that for the header level.

7.6.2 Adding a New Header Field into the CSV File

Custom fields can be written into the CSV file by adding script into *UserExitCSVFile*. To do this:

1. Open the project in the Designer application.
2. Switch to Definition Mode.
3. Right-click the **UserExits** class and select **Show Script** from the context menu.
4. In the script window, navigate to the user exit subroutine named **UserExitCSVFile**.
5. To add custom field **InvoiceCode** into the CSV file, add the following line into the subroutine:

   ```
   fnWriteCSVField(strRecordText, strKey, "<%ZIC>",
   pWorkdoc.Fields("InvoiceCode").Text)
   ```

   The *fnWriteCSVField* function has been specially provided for this purpose in order to condense the operation into a single command. In this case, **<%ZIC>** is a user-defined literal. It is recommended to use a **Z** as the first character of the literal so that there will be no conflicts with any standard system literals. **<%ZIC>** should subsequently be included in the CSV group line format parameters so that it is written into the file, for example:

   ```
   CSV_VL_01_FormatLine1=<%ZIC>
   ```

6. Close the script and save the project.

Note: If the CSV field to be added is a date field, and that date field is in the Verifier output format configured in the *DAT section* of the project configuration, then function *fnWriteCSVDateField* should be used instead. This function has an identical interface, but will convert the date into the date format configured for the CSV file group. If no group format has been set, the date will be formatted according to the output date format setting in the *EXP section* of the project configuration.

7.6.3 Adding a New Line Item Field into the CSV File

Custom line item fields can be written into the CSV file by adding script into *UserExitCSVFileLine*. To do this:

1. Open the project in the Designer application.
2. Switch to Definition Mode.
3. Right-click the **UserExits** class and select **Show Script** from the context menu.
4. In the script window, navigate to the user exit subroutine named **UserExitCSVFileLine**.
5. Add the appropriate custom script. This user exit works the same way as *UserExitCSVFile* described above, except that the *LineData* structure containing the current line item to be written into the file is included in the interface.
6. Close the script window and save the project.

7.6.4 Available Literals for Configuring the CSV Format

The following table details all of the available literal entries that can be used in the project configuration file when configuring the CSV output file format and their corresponding field or value.

CSV Section Parameter	Literal	Field / Value
NN_FormatLineN	<%TNM>	Name of the original imported file (no file path)

CSV Section Parameter	Literal	Field / Value
	<%TNF>	Name of the original imported file (no file path or file extension)
	<%TND>	Name of the original imported file with export directory file path
	<%SDT>	Scan date in the format YYYY-MM-DD
	<%BNM>	Batch Name
	<%EID>	Employee ID
	<%EFN>	Employee First Name
	<%ELN>	Employee Last Name
	<%DTP>	Document Type
	<%ITP>	Invoice Type
	<%IDT>	Invoice Date
	<%INO>	Invoice Number
	<%TAX>	Invoice Tax Amount
	<%WTX>	Invoice Withholding Tax Amount
	<%DCT>	Invoice Discount Amount
	<%MSC>	Invoice Miscellaneous Charge Amount
	<%CUR>	Invoice Currency
	<%TOT>	Invoice Total
	<%PST>	PST/QST Amount
	<%HST>	HST Amount
	<%ICM>	ICMS Tax Amount
	<%PON>	PO Number
	<%URN>	Unique Reference Number
	<%VID>	Vendor ID
	<%IVD>	Internal Vendor ID
	<%SID>	Site ID
	<%VNM>	Vendor Name
	<%BTO>	Bill-to Name
	<%CCO>	Company Code
	<%POR>	Payment Order Reference (POR) Number
	<%PSN>	Payment Order Subscriber Number
	<%KID>	Payment Reference
	<%ACC>	Account Number
	<%BAC>	Bank Account Number
	<%BCD>	Bank Account Code
	<%PRI>	Priority Flag
	<%EXC>	Exchange Rate
	<%IVR>	Invalid Reason
	<%ICD>	Invalid Reason Code
	<%LNK>	Document Link

CSV Section Parameter	Literal	Field / Value
	<%ERP>	ERP Document Key
	<%EPT>	ERP System PO Type
	<%BSU>	Business Unit
	<%DEL>	Delivery Note
	<%DLD>	Delivery Date
	<%DUE>	Due Date
	<%ISR>	ISR Retention Amount
NN_LineItem	<%LNO>	Invoice Line Item Number
	<%LPO>	PO Number
	<%LPL>	PO Line Item
	<%LDS>	PO Line Description
	<%LMN>	Material Number
	<%LMG>	Material Group
	<%LQT>	Quantity
	<%LUM>	Order Unit of Measure
	<%LUP>	Unit Price
	<%LPU>	Order Price Unit of Measure
	<%LQU>	Quantity in Order Price Unit of Measure
	<%LTO>	Line Item Total
	<%LTC>	Tax Code
	<%LTJ>	Tax Jurisdiction Code
	<%LFV>	Freight Vendor ID
	<%LGN>	Goods Receipt Document Number
	<%LGY>	Goods Receipt Document Year
	<%LGI>	Goods Receipt Document Item Number
	<%LSN>	Service Entry Sheet Number
	<%LSI>	'Service Entry Sheet Item Number
	<%LSD>	Subsequent Debit/Credit Indicator
	<%LLT>	Line Type
	<%LCD>	Charge Code
	<%LCI>	Charge Code ID
	<%LDL>	Distiller Line Item
	<%LPT>	Plant
	<%LTY>	ERP Purchase Order Type
	<%LBU>	ERP Purchasing Business Unit
	<%LCC>	Company Code
	<%LTR>	VAT / Tax Rate

7.7 Setting up a Custom Export

The following sections describe how a custom export may be implemented.

7.7.1 Introduction to Custom Exports

If data export is required in a format for which the existing export options cannot be leveraged to produce, or export is required for a custom base class, then a custom export is required.

The custom export must be scripted, and this is executed within a special user exit called **UserExitCustomExport**.

The table below described the parameters that are passed to the *UserExitCustomExport* subroutine:

Parameter Name	Description
pWorkdoc	Standard *Workdoc* object that provides access to all document field information (including the originally extracted line item data), the document classname, the document OCR text and the document filename.
ExportPath	Destination folder for file output. This value is taken from the export filepath configured on the RTS instance responsible for document export. If the RTS instance path is blank, then the export path will be set to the value held against the project configuration parameter *EXP_VL_DefaultExportPath*.
strDocLink	Path to the image of the document, which could be stored in either a storage directory or the batch directory; or it could be a URL to retrieve the image from an archive.
LineData	This multi-line array contains the line items available for export. This will be populated for all documents classified as invoices where line items are relevant.
GLData	Array based on the accounting data type defined in the *GlobalVariables* script level. This array will be populated if the system determines that general ledger coding entries are required for the document being processed. In the standard solution, this will only be populated for invoices where a miscellaneous charge was extracted at either header or line item level that is configured to be posted to a GL account in the *MSC section* of the project configuration.
TaxData	Array containing the total tax amount that corresponds to each tax code determined during the automatic tax calculation procedure.
blLinesRequired	Flag indicating whether line item export is relevant for the invoice based on the invoice characteristics, any invalid reasons set and the configuration in the *TAB section*. If set to **TRUE**, line items should be exported.
Address	Vendor address structure. This contains the address details for the document vendor.
Flags	Document validation flags. This structure contains document-specific flags, which can be used to determine what data should be exported.

The script contents of the user exit can be set to anything that is required.

It is advisable to check the document class before developing any script that refers to fields using hard-coded field names, particularly if the project uses custom base classes. If a field is referenced that does not exist in the document class, then a runtime error will occur. The global function *fnGetBaseClass* is provided for this purpose.

To activate the custom export, the following parameter in the *EXP section* of the project configuration must be set to **YES**:

```
EXP_OP_CustomExport=YES
```

If an error occurs during export, then populating the global variable *strExportError* with a descriptive error message will have the effect of failing the export and setting the document to state 750.

The user exit is called once for each document that is exported. Once export has been successful, the export history is updated against the document so that it will not be exported again by accident. Setting the document state to 200 will clear this history.

If the export is not successful, the user exit will be called again at the next attempt to export the document.

The history check can be overridden by setting the following parameter to **YES** in the project configuration:

```
EXP_OP_RedoAllExports=YES
```

Note: In production environments, it is recommended that this parameter be set to **NO**.

7.7.2 Considerations for Custom Export of Header Data

When using the custom export user exit for documents classified to the *Invoices* class, special care needs to be taken with the following header fields:

Header Field(s)	Comments
AmountTotal	When exporting the total invoice value, care must be taken to add back on the invoice withholding tax amounts. The correct script for calculating of the invoice total is: `dblTotal = fnConvertToDouble(pWorkdoc.Fields("AmountTotal").Text) + fnConvertToDouble(pWorkdoc.Fields(VAmountWithholdingTax").Text) + fnConvertToDouble(pWorkdoc.Fields("ISRRetention").Text)` If automatic tax determination is being used for countries that use tax jurisdictions, the *blShortPay* global variable will be set to **True** if the invoice total amount needs to be exported with the tax taken off. When writing amount fields into a text file or a database, the developer must also consider the locale upon which the project file is running. If, for example, *dblTotal* has a value of **100.54**, and the operating system locale uses a comma as the decimal separator then *CStr(dblTotal)* will be equal to **100,54**. To format the output of *dblTotal* consistently to use a period/full-stop as the decimal separator, the following line of script should be used: `strOutputTotal = Replace(CStr(dblTotal), ",", ".")` Amount fields may not be needed if the user has selected an invalid reason of **INVOICE AMOUNTS DO NOT ADD UP** in Verifier, as the content of the field may be incorrect. The *Invalid* property of the *Flags* object passed into the user exit provides a simple means to detect this, for example: `' Set total amount field for output` `Dim strOutputTotal As String` `If Flags.Invalid Then` `' amounts have been confirmed invalid in Verifier` `' invalid reason of 'INVOICE AMOUNTS DO NOT ADD UP' has been` `' set downstream system cannot accept amounts out of balance` `strOutputTotal = "0.00"`

Header Field(s)	Comments
	Else ' amounts confirmed in Verifier strOutputTotal = Replace(CStr(dblTotal), ",", ".") End If
AmountTax	Within the project file, four tax amount fields exist, which are: ▪ AmountTax (for total sales/use tax, VAT tax, Brazilian IPI tax, and the GST/HST component of Canadian tax) ▪ PST (Canadian PST/QST tax component) ▪ HST (Canadian HST tax component, if required separately to GST) ▪ ICMS (Brazilian ICMS tax). During export, all four should be summed together and passed as the total document tax, as shown in the following example script: dblTax = fnConvertToDouble(pWorkdoc.Fields("AmountTax").Text) + fnConvertToDouble(pWorkdoc.Fields("PST").Text) + fnConvertToDouble(pWorkdoc.Fields("ICMS").Text) + fnConvertToDouble(pWorkdoc.Fields("HST").Text)
AmountFreightPrepaidAndAdded Amount Misc	At time of export, the *AmountFreightPrepaidAndAdded* and *AmountMisc* fields will only contain the miscellaneous charges that were extracted, or entered by a user, at header level. Exporting these values may provide an incomplete picture if miscellaneous charges were also captured at line item level. Even though miscellaneous charges may appear on the invoice at header level, from the system point of view, they are considered part of the line item data. Hence, miscellaneous charges should only be exported if the document is relevant for line items, which can be determined if the user exit import parameter *blLinesRequired* is set to **True**. The correct way to export miscellaneous charges depends on how the output type has been configured in the *MSC section* of the project configuration. Of the standard output types, only the option to book the charge as an unplanned delivery cost is relevant for a header level field export. **Example 1:** All miscellaneous charges have been configured to post to as unplanned costs, which means that the system will sum up all identified charges (whether presented on the invoice at header or at line item level), and this total value will be available in global variable *dblUnplannedDelCost*. **Example 2:** The customer has three types of miscellaneous charges: freight, pallet charges and customs charges. In the *MSC section* of the project configuration, three groups have been set up to represent these charges (with codes **F**, **P** and **C** respectively) and have been configured to extract the data correctly. The client wishes to pass the sum total of all of these charges by charge type at header level to their downstream system using a custom export. As this is a non-standard output, no processing action has been set against any of the miscellaneous charges. In the custom export user exit, the sum total of all charges can be retrieved into local variables via the following script: Dim dblTotalFreight As Double Dim dblTotalPallet As Double Dim dblTotalCustom As Double If blLinesRequired Then dblTotalFreight = fnGetMiscChargeTotalForCode(pWorkdoc, "F") dblTotalPallet = fnGetMiscChargeTotalForCode(pWorkdoc, "P") dblTotalCustom = fnGetMiscChargeTotalForCode(pWorkdoc, "C")

Header Field(s)	Comments
	`End If` If the above script is used and line pairing is switched off, but line items are still needed for export, then line items should not be exported from the *LineData* array if *LineData(RowNum).MiscCharge* = **X** as they will already have been encountered for.
AmountSubtotal	Under the standard configuration, the subtotal is not a required field. Therefore, the standard project field may not contain the correct value for export. For the correct calculation of the subtotal, which is, by definition, the amount before sales tax/VAT, the following script should be used: `dblSubtotal = dblTotal – dblTax` See above for the correct calculations of *dblTotal* and *dblTax*.
VendorID	The correct way to export the vendor ID depends on what vendor ID is required by the downstream system. The following examples show the correct vendor IDs to be exported for different cases. **Example 1:** ERP uses an internal vendor ID, a site ID and an external vendor ID. The internal vendor ID is held in *pWorkdoc.Fields("InternalVendorID").Text* The site ID is held in *pWorkdoc.Fields("SiteID").Text* The external vendor ID is held in *pWorkdoc.Fields("VendorID").Text* **Example 2:** ERP uses a vendor ID and a site ID. The vendor ID is held in *pWorkdoc.Fields("VendorID").Text* The site ID is held in *pWorkdoc.Fields("SiteID").Text* **Example 3:** ERP just uses a regular vendor ID. The vendor ID is held in *pWorkdoc.Fields("VendorID").Text* In all cases, the unique vendor ID is held in the *ID* component of the *Address* structure. If the export requires writing out elements of the vendor address, then these attributes are available in the *Address* structure passed into the user exit. For example, the following script retrieves the vendor name: `Dim strName As String` `strName = Address.Name`
PONumber	Purchase order numbers are usually exported at line item level. Each invoice could potentially refer to multiple purchase orders, and it is the job of the line pairing operation to work out which of those purchase orders and which line item on those purchase orders each invoice line relates to. However, from the Verifier point of view, only one purchase order number is required at header level to be able to validate a document. If multiple purchase orders are present on the document, then this is handled at time of document export. If line pairing is not being used, or if the downstream system requires a single purchase order number to be passed at header level, then this may be retrieved from the standard *PONumber* field. In Verifier, it is possible that a user selected a **Missing/Invalid PO** or **Missing/ Invalid Vendor AND PO** invalid reason, so the content of the field may not be correct. The *NOPO* property of the *Flags* object provides an easy way to detect this. For example: `Dim strOutputPO As String` `If Flags.NoPO Then`

Header Field(s)	Comments
	`` ' User selected an invalid reason, so the PO may be wrong ' Don't pass the PO strOutputPO = "" Else strOutputPO = pWorkdoc.Fields("PONumber").Text End If ```
ExchangeRate	The exchange rate is exported at header level, and is typically populated in instances where the vendor is charging value added tax, but not in the local currency of the country where VAT is levied. For example: A US vendor invoices a UK company. As the US company does a significant amount of business in the UK, they are legally obliged to be UK VAT-registered and to charge VAT on their invoices. The US company presents its invoice is US Dollars. The law requires them to specify the VAT in British Pounds as well as in US Dollars. Alternatively, they may quote an exchange rate instead. Within the Verifier application, depending on whether the vendor has chosen to quote an exchange rate or tax amount in GBP, user input could be in either of the two fields, but the export value needs to be a specific exchange rate. Simply passing the content of the *ExchangeRate* field may be insufficient. To assist with this, global function *fnCalculateExchangeRate* is available. This will format and pass an exchange rate if entered in the exchange rate field, and will also back-calculate the exchange based on the values in the *AmountTax* and *LocalVATAmount* fields, if those were populated instead. If both fields are blank, or contain zero or invalid values, an empty string will be returned. All exchange rates passed back will be formatted to 5 decimal places. The below script shows a sample usage of this function: ``` Dim strExchangeRateForExport As String strExchangeRateForExport = fnCalculateExchangeRate(pWorkdoc) ```

7.7.3 Custom Export of Line Item Data

Line item data for export is contained within the *LineData* array, which is a 1-based array.

No line items with a value of zero will be included within the array, except in a very special case where line pairing is enabled and the invoice is third-party freight with all miscellaneous charges set to be posted as unplanned costs. In this instance, a single zero-value line will be written into the array, but with the subsequent credit/debit flag set to **X**.

The contents of the *LineData* array will vary depending on whether line pairing is enabled or disabled. If line pairing is enabled, the array will contain:

1. All material/service line items that the system was able to pair.
2. All material/service line items that the system was unable to pair.
3. All miscellaneous charges that were set to post as a specific line type.
4. All miscellaneous charges where no processing option has been selected.

Paired lines can be identified by checking the value of the *PO_Number* and *PO_Item* properties of the line item. If they are populated, the line was paired, but if they are empty, it was not paired. Miscellaneous charges set to post as a specific line type can be identified by checking the value of the *MiscCharge* property of the line item. In this instance, that property will have a value of **X**.

For miscellaneous charges that were set to be booked against a general ledger account entry, these can be found in the *GLData* array. For miscellaneous charges which were set to be booked

as unplanned costs, these can be found summed together in the global variable *dblUnplannedDelCost*.

If line pairing disabled, the array will contain:

1. All material/service line items captured from the invoice.
2. All miscellaneous charges captured at line item level from the invoice.

Miscellaneous charges can be differentiated from regular material/service line items as the *MiscCharge* property of the line item will contain **X**.

Not all documents require line items to be exported, either because they are not relevant for the type of invoice, or because a user selected a particular invalid reason in the Verifier application, or because they are not required by the implementation in general. Settings in the *TAB section* of the project configuration control the types of document and circumstances where line item extraction is relevant.

Within *UserExitCustomExport*, the Boolean import parameter *blLinesRequired* provides a simple indicator as to whether line items need to be exported for the current document.

The following script illustrates basic export of line item information into a flat file with a **LNS** extension:

```
Dim lngLine As Long
Dim Line As LineData
Dim strQuantity As String
Dim strTotal As String
Dim strDescription As String

' Check whether document is relevant for line level export
If blLinesRequired Then
   Open ExportPath & fnGetFileName(pWorkdoc.DocFileName(0)) & ".lns" For
   Output As #1

   For lngLine = 1 To UBound(LineData)
      Line = LineData(lngLine)
      ' Format amount components to ensure that a full-stop/period will
      ' always appear as the decimal separator in the output file
      strQuantity = Replace(CStr(Line.Quantity), ",", ".")
      strTotal = Replace(CStr(Line.Item_Amount), ",", ".")

      ' Format description to remove commas and limit to 40 chars
      strDescription = Left(Replace(Line.Description, ",", ""), 40)

      ' Export PO details, description, quantity and total
      Print #1, Line.PO_Number & "," & Line.PO_Item & "," & _
                strDescription & "," & _
                strQuantity & "," & strTotal
   Next lngLine

   Close #1

End If
```

Note: If the invoice is a Brazilian Nota Fiscal with ICMS tax, the line items held within the *LineData* array will have unit prices and totals exclusive of ICMS tax, even though the line items appear on the invoice inclusive of ICMS tax.

8 Improving Data Extraction

8.1 Introduction

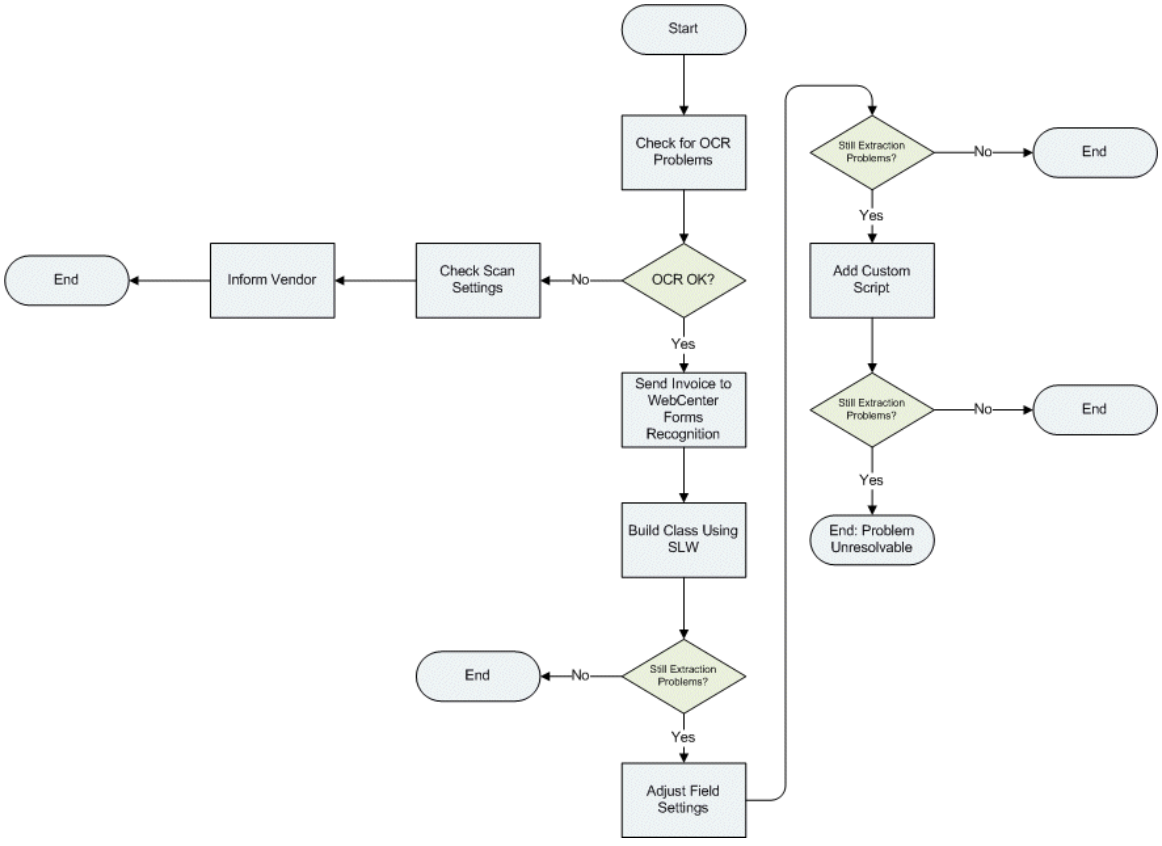
The AP Packaged Project comes pre-configured with a generic learnset for the extraction of data from invoices. This means that, out-of-the-box, the system should already be able to deliver industry-leading extraction results from invoice documents of any format without any special configuration.

However, for those documents which stop in the Verifier application due to missing or incorrectly read fields, it is possible to take additional measures to build upon the system's existing extraction capability, thus ensuring that these problematic documents pass through the system untouched.

The following sections describe the process that should be followed in order to achieve this.

8.2 Extraction Improvement Process Flow

The diagram below shows the process that should be followed in order to improve extraction for each test case identified:



8.2.1 Checking for OCR Problems

Problems with the OCR read of a document are the chief cause as to why the AP Packaged Project may not be able to extract the invoice information. When considering the viability of improving extraction for a given invoice, the first thing to check is the document OCR results.

There are two types of OCR problems that affect the generic extraction results:

1. OCR problems that affect the field value itself that requires extraction.
2. OCR problems that affect the context surrounding the field value.

If the document OCR is particularly poor, to the point where the information that required extracting was compromised, then the document is not a good candidate for further improvement.

The following sections describe the two different varieties of OCR error that may occur, and suggest possible courses of action that may be taken.

8.2.1.1 OCR Problems on the Field Value

OCR problems with the field value itself may be caused by a number of factors relating to the original document, including (but not limited to):

1. Handwriting or stamps obscuring the required data.
2. Poor document print quality.
3. Shading on the document, for example black lettering on a dark grey background.
4. Misalignment of print text against a stationery background.
5. Handwritten information.

Checking the invoice OCR read can be done in both the Designer and Verifier applications by hovering the cursor over the area of the document where the field value is located.

In Designer, when viewing a document, clicking the *Highlight All Words* button on the top toolbar will show the OCR results for the entire document. Each highlighted block on the image represents a single OCR word. Positioning the cursor over each highlighted block shows how the system has interpreted the word.

There is very little that can be done to overcome problems when poor OCR has compromised the actual field value itself, but some possible options include:

1. Adjusting scanner settings using technologies such as Virtual Rescan or PerfectPage, which can prove beneficial for problems caused by shading and text misalignment.
2. Contacting the vendor to request a better quality of invoice.

8.2.1.2 OCR Problems on the Field Contextual Information

Poor OCR on the contextual information surrounding the field, or no contextual information at all, can also compromise generic field extraction. The extraction engine uses this context to assign confidence values to field candidates, so, if the context is not available or it cannot be read correctly, the correct candidate may not be accorded to requisite level of confidence to be extracted.

Note: The OCR on the context does not need to be perfect as the system uses fuzzy pattern-matching technologies to interpret the information surrounding the field.

The absence of a meaningful context does not mean that extraction failure is guaranteed, as the system will also consider the value in relation to other fields, along with its general position on the document. However missing contextual information will lead to the correct candidate being accorded a lower confidence value in comparison to documents where a context was provided.

Extraction issues that fall into this category can be remedied by creating a class using the Supervised Learning Workflow, which is described in the next section.

8.2.2 Building a Class Using Supervised Learning Workflow

The WebCenter Forms Recognition Supervised Learning Workflow is a standard core product feature that allows users to train the system to improve extraction for problematic invoices in a production environment.

It is appropriate for use in instances where field data is not being extracted due to low system confidence or in instances where the system is delivering an incorrect extraction result. It cannot remedy problems caused by poor OCR, except in instances where the OCR issue concerns the surrounding field contextual information.

Before using the Supervised Learning Workflow, the Learnset Manager user must consider whether this is a worthwhile step for the invoice in question. For example, if very few invoices are received from the vendor in question, then it may not be worth creating a class, instead letting the documents stop in Verifier. However, if the invoice is from a high-volume vendor, then creating a class makes much more sense.

Oracle recommends a limit of 500 vendor subclasses classes per project file.

In a production environment, learnset additions are proposed by users through the standard Verifier application. The additions are subsequently reviewed by a learnset manager, who subsequently decides which of those additions are suitable to be promoted to the global project.

In the background, this learnset addition creates a separate vendor subclass that contains the learnset improvement. The format of the name of the new class is derived from the settings against the *VendorASSA* field on the *Invoices* class level, namely the *Class Name Format* setting. Typically, this is set to the vendor number, then an underscore, then name of the vendor, although the class is actually used for any invoice of a similar layout that exhibits a similar extraction problem.

Further information regarding the naming convention for vendor subclasses can be found in *Appendix D: Configuring the Vendor ID Field*.

8.2.2.1 Using Supervised Learning Workflow in Designer

It is possible to use the Supervised Learning Workflow feature within the Designer application, which is done using *Verifier Train Mode*. This process is recommended for a system administrator, as it shortcuts the process of having to approve the invoice using the Learnset Manager application.

To use *Verifier Train Mode* for learning in Designer the following pre-requisites must be in place:

1. Supervised learning must be activated in the project file. This is done on the **Verifier Train Mode** tab in the project settings.
2. The *Classification Field* setting on the **Document Class** tab of the *Invoices* class-level properties should be set to **VendorASSA**.
3. A Verifier form must be defined against the *Invoices* class.

4. The project should be pointing to the relevant Learnset Manager *Base Directory*. This is specified on the **Train Mode** tab of the project settings.
5. The data pool for the *VendorASSA* field must be created and ready.

When all of the above prerequisites have been fulfilled, it is possible to train a document using Supervised Learning Workflow. The steps to achieve are as follows:

1. Navigate to the problem document in the batch.
2. Analyze the document in the *Fields* view of *Definition Mode*.
3. Switch to *Verifier Train Mode*.
4. Ensure that the **Add to Learnset** button is depressed.
5. Complete verification of the document.

   When completing header data verification, the user must be sure to click on the correct text on the image of the document to copy it into the field so that the system knows where the data was selected on the document. The new class will not be trained properly if the data is simply keyed in manually.

   Important Note: If the document requires line items to be extracted, these must be trained using the Table Correction tool, even if they were extracted 100% using the generic line item extraction.

   If a field contains an incorrect extraction result and the value is not actually present on the document, the user should navigate to the area of the document where the incorrect extraction result was found, right-click on it, and select *Clear Candidate Assignment* from the context menu.

   When document verification has been completed, pressing the [Enter] key on the last invalid field will move the system to the next document in the batch.
6. Click the **Learn Documents** button to train the class.
7. Save the project file.

Navigating back to the class tree in *Definition Mode* at this stage shows the new class that has been created. At this point, the new class should be tested, initially with the document used to create the class, and then with further documents from the same vendor or of the same format.

If extraction problems still persist then the next step is to examine the class field settings.

8.2.3 Adjusting Field Settings

Adjusting field settings can overcome many extraction problems encountered subsequent to building a class using the Supervised Learning Workflow. Changing field settings encompasses:

1. Adjusting candidate format strings and definitions.
2. Adjusting field confidence and distance values.

Below are some example problems that may be encountered, along with suggested solutions.

Example 1: Too few characters of the invoice number are being extracted for the new subclass.

In this example, the invoice number stated on a German invoice is **2009-06-0-00510**, but the new subclass is extracting just the latter part of the number (i.e. **00510**) as the field result.

This is occurring is because the format string for the invoice number, which the new subclass has inherited from the field definition on the *Invoices* class level, is insufficient to extract the invoice number in its entirety.

The corrective course of action is, therefore, to adjust the format settings for the invoice number on the new subclass. The steps to achieve this are as follows:

1. Go to the class view screen and double-click on the new subclass to see the fields.
2. Navigate to the *Invoice Number* field and show the field's properties.
3. From the dropdown of *Available Templates*, select the **InvoiceNumber** template.
4. Click the **Copy Template** button.

   The generic format settings for the *Invoice Number* field are displayed. The user is free to manipulate these settings in any way deemed appropriate as they only apply to the new document subclass, and will not have a wider effect on the project. Under no circumstances should the format settings against fields on the *Invoices* class ever be changed; doing so may have a catastrophic effect on the generic extraction results.

   The generic *Format String* is configured as **?[2-16]** which means that a valid candidate for the invoice number is permitted to be any sequence of alpha or numeric characters between 2 and 16 in length. In the *Ignore Characters* field, the characters that may also be permitted to appear in an invoice number candidate in any position are defined.

   Note: The invoice number given in this example scenario is 16 characters in length, so these format settings are appropriate and do not require changing.

5. Next, click on the **General** tab to show the general settings for the field.

   Notice that the *Max. Wordcount* field (typically) has a value of **5**. This means that the invoice number is permitted to be a maximum of five OCR words. In this example, the desired invoice number consists of 7 OCR words, where each number string and each hyphen character are considered individual words.
6. In this example, the correct course of action is to increase the *Max. Wordcount*' property value to **7**.
7. Save the project.

With these new settings, the invoice number will now be extracted correctly for the subclass, without jeopardizing the extraction results for any other document classes.

In instances where the word count is already sufficient, it could be the case that the distance between the words is causing the problem in the sense that it is too small. For the invoice number, the default distance inherited from the *Invoices* class level is **2.5**, which is shown in the *Max. gap between words* parameter. Increasing this distance can help solve this problem.

Example 2: The system keeps extracting a value for the tax, but this vendor never charges tax

This problem tends to occur if the class was created incorrectly, and the *Clear Candidate Assignment* option was not used to reset an incorrect tax extraction, as described in step 5 of

Section 8.2.2.1: Using Supervised Learning Workflow in Designer above. The steps to correct this are as follows:

1. Go to the class view screen and double-click on the new subclass to see the fields.
2. Navigate to the *AmountTax* field and show the field's properties.
3. From the dropdown of *Available Analysis Engines*, select the **Format Analysis Engine**.
4. Configure no format strings. This means that no candidates can ever be generated for the tax amount hence no value can ever be extracted.
5. Save the project

8.2.4 Add Custom Script

Adding custom script should always be the last resort for the correction of extraction problems. Each individual vendor learnset has its own script class, which means that code can be added without interfering with the operation of any other part of the project.

Example usages of script include:

1. Correcting OCR problems where the correct result will always be known for a given vendor.
2. Defaulting mandatory field values that the vendor does not state on the invoice.
3. Improving line item extraction for vendors who present the information in a way that is not entirely supported by the Table Extraction Engine.

Example 1: Script correction of a known OCR issue with the invoice number

The vendor presents an invoice number that is always suffixed with **RI** but the OCR engine always reads it as **RL**. Consequently, documents from this vendor are always stopping in Verifier for user correction.

Script may be added to the *InvoiceNumber_Format* procedure to clean up the OCR misread.

Example 2: Defaulting a price unit in the table of line items

The vendor submits invoices where a price unit of **100** is always missing from the line item detail on the document. The totals, quantities and unit prices are being read correctly, but do not validate mathematically because of the absence of a price unit. Hence, all documents are stopping in Verifier for the user to key **100** into the *Price Unit* column for each line item so that the document can pass.

To handle this issue, custom script can be inserted into the *LineItems_PostEvaluate* event against the class to automatically populate the *Price Unit* column for each line item with the desired value.

Example 3: Applying custom formatting to an invoice date

A vendor always presents the invoice date on the document in the format **YY MM DD**, which the system often is unable to translate. For example, **10 09 09** could mean 9th September 2010 or 10th September 2009 from the system point of view. As a result, documents are stopping in Verifier for the user to key in the date in a **DD/MM/YYYY** format.

Custom script can be inserted into the *InvoiceDate_Format* event for the vendor class in order to automate this.

Appendix A Tax Configuration for Countries Without Tax Jurisdictions

A1. Introduction to the Tax Table

The AP Packaged Project uses a lookup table for determining and validating tax codes for purchase order based invoices, prior to posting them in the downstream ERP system. It is appropriate for use with countries where the tax rate is determined by the actual tax code itself, rather than an associated tax jurisdiction code as used in the US, Canada and Brazil.

The tax table is installed into either an Oracle or SQL Server database using an install SQL script that is provided with the AP Packaged Project. The script file is located at:

<*Installation Folder*>**\Projects\AP 1007G\DB Scripts\WFR_AP_Tables_Create_**<*database*>**.sql**

Usage of the tax table is activated in the *TAX section* of the project configuration.

A2. Structure of the Tax Table

The columns in the tax table are as follows:

Column Name	Type	Description
COUNTRY	String	Country code for the country to which the tax code applies. It represents the country in which the company code is legally registered. For example: GB, FR, CH, DE, etc. This column must be populated for all records in the table.
TAXCODE	String	Tax code.
SHIPTO	String	Ship-to country code. This is the country code for the country to which the goods were shipped. This is compared by the system to either the country of the company that received the goods, or the country of the plant set on the relevant purchase order line item depending on the configuration specified in the *TAX section*. If the ship-to country is the same as the company code country, then it should be entered as it is. If it is different, but is an EU member state, it should be set to **E**'; if it is a non EU-member state, it should be set to **XX**. This column must be populated for all records in the table.
SHIPFROM	String	Ship-from country code. This is the country code for the country from where the goods were shipped. This is compared by the system to the country of the order-from vendor. If the ship-from country is the same as the company code country, then it should be entered as it is. If it is different, but is an EU member state, it should be set to **EU**; if it is a non EU-member state, it should be set to **XX**. This column must be populated for all records in the table.
SERVICE	'X' or blank	This denotes that the record in the tax table is relevant for a service, as opposed to a material line item.
MATERIALGROUP	String	Purchase order material category/type/group or class.
MATERIALNO	String	Purchase order material number.
VENDORID	String	Order-from vendor ID.

Column Name	Type	Description
PERCENTAGE	Number (two decimal places)	Percentage rate associated with the tax code (e.g. 17.5, 19.6, 10, 18 etc.) This value will be compared with the tax rate captured at the line item level to support mixed tax rates on a single invoice, or, if no rate is captured, then it will be compared to the overall tax rate for the invoice as a whole. When populating the tax table, **999** denotes a blank or unknown percentage. If the percentage is zero, then **0** should be entered.

A3. Tax Table Access Sequence

At runtime, the AP Packaged Project uses an access sequence to interrogate the tax table in order to find the correct tax code for each invoice line.

This step will not be carried out if the system is configured to use the purchase order line tax code in all cases and that tax code is populated or if the tax table is not populated with any entries relating to the country of the invoice company code.

The access sequence is as follows:

1. Check for combination of vendor, material group, ship-to and ship-from
2. Check for vendor and combination of ship-to and ship-from
3. Check for material number and combination of ship-to and ship-from
4. Check for material group and combination of ship-to and ship-from
5. Check for combination of percentage, ship-to, ship-from and service indicator
6. Check for combination of ship-to, ship-from and service indicator
7. Check for combination of ship-to, ship-from and tax percentage
8. Check for ship-to and ship-from combination (default tax code for the country)

The system begins at step 1. If a tax code is found for that step in the access sequence, the corresponding tax code is used; if not, the system moves to step 2, and so on.

If, at the end of step 8, no tax code can be determined, then the invoice will go to exception for reasons of a missing tax code.

A4. Populating the Tax Table

The following table shows an example of how the tax code table might typically be populated for the UK (Country Code = GB):

COUNTRY	TAX CODE	VENDOR ID	SHIP TO	SHIP FROM	SERVICE	MATERIAL GROUP	MATERIAL NO	PERCENTAGE
GB	V0		GB	GB				0
GB	V1		GB	GB				17.5
GB	V3		GB	EU				999
GB	V4		GB	EU	X			999
GB	V6		GB	GB				8
GB	V9		GB	GB				15
GB	VF		GB	XX				999

COUNTRY	TAX CODE	VENDOR ID	SHIP TO	SHIP FROM	SERVICE	MATERIAL GROUP	MATERIAL NO	PERCENT AGE
GB	V9		GB	GB				999

Using this configuration, the system will behave as follows as a result of each record in turn:

1. If it was a domestic transaction, and no tax was charged, then tax code **V0** will always be used
2. If it was a domestic transaction, and tax at the rate of 17.5% was charged on the invoice, then tax code **V1** will always be used
3. If the goods were shipped from an EU member state country to the UK, then tax code **V3** will always be used
4. If the invoice is from an EU member state vendor and the service was performed in the UK, then tax code **V4** will always be used
5. If it was a domestic transaction, and tax at the rate of 8% was charged on the invoice, then tax code **V6** will always be used
6. If it was a domestic transaction, and tax at the rate of 15% was charged on the invoice, then tax code **V9** will always be used
7. If the invoice is from a vendor outside of the EU, and the goods were shipped to the UK, then tax code **VF** will always be used
8. If no specific item tax percentage was determined from the invoice, but it was a domestic transaction, then tax code **V9** will always be used

Step 8 represents the default tax code for a combination of ship-to and ship-from countries. The most common non-zero tax code should be set against this record.

If an additional business requirement involved a new tax code (**VS**) that was to be used in instances where a non-EU member state vendor submitted an invoice for a service carried out outside the EU where tax was charged at 0%, then the following entry should be added to the tax table:

COUNTRY	TAX CODE	VENDOR ID	SHIP TO	SHIP FROM	SERVICE	MATERIAL GROUP	MATERIAL NO	PERCENT AGE
GB	VS		XX	XX	X			0

Equally, if a requirement arose whereby a different tax code (**DS**) was to be used in all instances involving a service carried out domestically irrespective of whether tax was charged or not, the following entry should be added to the tax table:

COUNTRY	TAX CODE	VENDOR ID	SHIP TO	SHIP FROM	SERVICE	MATERIAL GROUP	MATERIAL NO	PERCENT AGE
GB	DS		GB	GB	X			999

A5. Handling Special Cases

The tax table provides functionality for handling special cases where a specific tax code is required.

This permits the user to:

1. Allocate a specific tax code to a specific vendor, irrespective of the type of transaction and the rate of tax charged
2. Allocate a specific tax code for a specific material irrespective of the rate of tax charged
3. Allocate a specific tax code for a specific material group irrespective of the rate of tax charged

This can be used, for example, in instances where the tax against an individual invoice is not VAT reclaimable, or only partially reclaimable.

For example, all goods and services provided by vendor 12345 are only 50% VAT reclaimable, so a 50% VAT reclaimable tax code (**VR**) should always be used.

The corresponding tax table entry should look as follows:

COUNTRY	TAX CODE	VENDOR ID	SHIP TO	SHIP FROM	SERVICE	MATERIAL GROUP	MATERIAL NO	PERCENT AGE
GB	VR	000012345	GB	GB				999

By creating entries that include a combination of the vendor, material number and the material group, these instances can be handled as the access sequence will check for such entries before selecting a tax code based solely on the type of transaction (e.g. service) and the tax rate.

It is also possible to leave the tax code column blank for such special cases, which will always force those invoices to exception in a downstream process for someone to review.

For all entries in the tax table relating to special cases, the ship-to and ship-from countries must continue to be populated.

Appendix B Tax Configuration for Countries Using Tax Jurisdictions

B1. Introduction

This appendix describes the steps required to complete the tax configuration section for ERP systems and countries that use tax jurisdiction codes.

If tax jurisdictions codes are used, the system requires two items per invoice line in order to be able to create the invoice successfully, specifically:

1. The invoice line item tax code
2. The invoice line item tax jurisdiction code

Under this ERP system configuration, which is applied at the country level (typical subject countries would be the US, Canada, India and Brazil where the tax rate is calculated based on the part of the country where the item was bought or consumed), the tax jurisdiction code represents the rate of tax applied to the purchase, and the tax code represents whether the item is taxable or non-taxable as well as serving as an instruction on how the ERP system should process the tax.

Often the rates connected to tax jurisdiction codes are held within a specialist 3rd party tax application external to the ERP system.

B2. Basic Configuration Example

The following is a simple configuration for a US-based example:

In this example, each line on the purchase order read from the ERP system has been allotted a tax code.

Code **I0** denotes a tax-exempt item; code **I4** denotes that the item is taxable.

When the invoice arrives, it is for a single line item. The line pairing routine successfully identifies that line item and the system proceeds to consider the tax code.

In this scenario, there are four basic possibilities:

- If the purchase order line has a tax code of **I0**, and no tax is charged on the invoice, the system should use tax code **I0** for the invoice line
- If the purchase order line has a tax code of **I0**, and the vendor is charging tax, the tax code should remain at **I0** for the invoice line, and the vendor should be 'short-paid' the tax they are billing (i.e. the total amount of the invoice should be the total amount minus the total tax amount)
- If the purchase order line has a tax code of **I4**, and no tax is charged on the invoice when tax is expected, the system should use a tax code for the invoice line that will self-assess the use tax for payment to local tax authorities (code U1).
- If the purchase order line has a tax code of **I4**, and tax is charged on the invoice, the invoice line tax code should remain at **I4**, and the tax should be paid to the vendor as billed.

The following tax group entries should be created in the *TAX section* of the project configuration:

```
TAX_VL_01_TaxCode=I0
TAX_VL_01_Country=US
TAX_VL_01_IfTax=
TAX_VL_01_IfNoTax=
TAX_VL_01_VendorState=
TAX_VL_01_ShipToState=
TAX_VL_01_IfTaxState=
TAX_VL_01_IfNoTaxState=
TAX_OP_01_PayTaxAsBilled=NO
TAX_OP_01_ShortPayIfTax=YES
TAX_VL_01_AccountAssignmentCategories=

TAX_VL_02_TaxCode=I4
TAX_VL_02_Country=US
TAX_VL_02_IfTax=
TAX_VL_02_IfNoTax=U1
TAX_VL_02_VendorState=
TAX_VL_02_ShipToState=
TAX_VL_02_IfTaxState=
TAX_VL_02_IfNoTaxState=
TAX_OP_02_PayTaxAsBilled=YES
TAX_OP_02_ShortPayIfTax=NO
TAX_VL_02_AccountAssignmentCategories=

TAX_VL_03_TaxCode=U1
TAX_VL_03_Country=US
TAX_VL_03_IfTax=
TAX_VL_03_IfNoTax=
TAX_VL_03_VendorState=
TAX_VL_03_ShipToState=
TAX_VL_03_IfTaxState=
TAX_VL_03_IfNoTaxState=
TAX_OP_03_PayTaxAsBilled=NO
TAX_OP_03_ShortPayIfTax=YES
TAX_VL_03_AccountAssignmentCategories=
```

The tax jurisdiction code passed to the invoice line item will always be set to the tax jurisdiction code set against the purchase order line item.

If the invoice has multiple line items, the vendor will only be short paid the tax if tax was being billed and all tax codes selected for the invoice lines carry a short-pay instruction.

If the invoice has multiple line items, the vendor is charging tax, and any single invoice line item tax code carries an instruction to pay the tax as billed, the system will book the entire invoice tax amount to the ERP system using the pay-tax-as-billed tax code in question.

B3. Defaulting a Purchase Order Tax Code

If no tax code is present on the purchase order line item, the system is able to default a tax code based on the purchase order line account assignment category.

To configure this, the appropriate account assignment categories should be added as a comma-separated list in the **AccountAssignmentCategories** parameter for the desired tax code. An asterisk denotes a blank account assignment category.

The new purchase order tax code selected by the process above is then subject to the logic in order to determine the invoice line tax code.

B4. Configuring a State-Dependent Tax Code

Should a specific tax code be required depending on the vendor or ship-to state, the following configuration options are available as illustrated by the following examples:

1. If the purchase order line tax code is **I4**, but this should change to **U1** if no tax is charged, except if the vendor is based in California, in which case code **IU** should be used, then this would be configured as follows:

```
TAX_VL_02_TaxCode=I4
TAX_VL_02_Country=US
TAX_VL_02_IfTax=
TAX_VL_02_IfNoTax=U1
TAX_VL_02_VendorState=CA
TAX_VL_02_ShipToState=
TAX_VL_02_IfTaxState=
TAX_VL_02_IfNoTaxState=IU
TAX_OP_02_PayTaxAsBilled=YES
TAX_OP_02_ShortPayIfTax=NO
TAX_VL_02_AccountAssignmentCategories=
```

2. If the purchase order line tax code is **I4**, but this should change to **U1** if no tax is charged, except if the ship-to state is Texas, in which case code **IU** should be used, then this would be configured as follows:

```
TAX_VL_02_TaxCode=I4
TAX_VL_02_Country=US
TAX_VL_02_IfTax=
TAX_VL_02_IfNoTax=U1
TAX_VL_02_VendorState=
TAX_VL_02_ShipToState=TX
TAX_VL_02_IfTaxState=
TAX_VL_02_IfNoTaxState=IU
TAX_OP_02_PayTaxAsBilled=YES
TAX_OP_02_ShortPayIfTax=NO
TAX_VL_02_AccountAssignmentCategories=
```

3. If the purchase order line tax code is **I4**, but this should change to **U1** if no tax is charged, except if the vendor and the ship-to were both in Virginia, in which case code **IU** should be used, then this would be configured as follows:

```
TAX_VL_02_TaxCode=I4
TAX_VL_02_Country=US
TAX_VL_02_IfTax=
TAX_VL_02_IfNoTax=U1
TAX_VL_02_VendorState=VA
TAX_VL_02_ShipToState=VA
TAX_VL_02_IfTaxState=
TAX_VL_02_IfNoTaxState=IU
TAX_OP_02_PayTaxAsBilled=YES
TAX_OP_02_ShortPayIfTax=NO
TAX_VL_02_AccountAssignmentCategories=
```

Note that if both **VendorState** and **ShipToState** are populated, then only that combination will trigger use of either the **IfTaxState** or **IfNoTaxState** codes. In example 3 above, if the vendor was from Virginia, the ship-to state was Maryland, the purchase order line tax code was **I4** and no tax was charged, tax code **U1** would be used.

The state tax code feature will not function unless the plant validation option is activated and pointing to a populated data source in the *TAX section* of the project configuration settings.

Appendix C Configuring the Invoice Type Field

The Invoice Type field represents whether the invoice is purchase order or non-purchase order related (**PO** or **NO-PO**), which, in turn controls whether the purchase order number and line item fields are mandatory, and how the document is handled at the point of export.

The project configuration determines the initial setting for this field, as well as how the circumstances under which this default setting should be changed.

The following sections show three typical example business requirements and the corresponding settings in the *ITY section*, *SRC section* and *IMP section* of the project configuration

Business Requirement 1

The invoice type should be **NO-PO** unless a purchase order is found on the document.

```
ITY_VL_Default=NPO
ITY_OP_SetByVendor=NO
ITY_VL_SetByVendorCCExceptions=
ITY_VL_POValue=MM
ITY_VL_NPOValue=FI
ITY_OP_SetToPOIfPOFound=YES
ITY_OP_SetToPOIfValidPOFound=NO
ITY_OP_SetToPOIfPOPopulated=NO
```

To switch the invoice type to **PO** not just based on whether a purchase order is found on the invoice, but also based on whether it exists in the PO validation database, the latter three parameters should be set as follows:

```
ITY_OP_SetToPOIfPOFound=NO
ITY_OP_SetToPOIfValidPOFound=YES
ITY_OP_SetToPOIfPOPopulated=NO
```

If the invoice type should always be set to **PO** if the purchase order number field has content, whether captured automatically or entered by a user in Verifier, the following parameter should always be set to **YES**:

```
ITY_OP_SetToPOIfPOPopulated=YES
```

Business Requirement 2

The invoice type should default to **PO**, but should switch to **NO-PO** if no purchase order number is found and the vendor is permitted to supply invoices without a purchase order.

```
ITY_VL_Default=PO
ITY_OP_SetByVendor=YES
ITY_VL_SetByVendorCCExceptions=
ITY_VL_POValue=MM
ITY_VL_NPOValue=FI
ITY_OP_SetToPOIfPOFound=YES
ITY_OP_SetToPOIfValidPOFound=NO
ITY_OP_SetToPOIfPOPopulated=NO

SRC_VL_InvoiceType=XXXXXX
```

In the SRC setting above, *XXXXXX* represents the name of the vendor ASSA field column to which the invoice type value is passed from the vendor extract file. The value of this component for both **PO** and **NO-PO** invoices must be mapped to the **ITY_VL_POValue** and

ITY_VL_NPOValue. In the example above, the system is expecting to find either **MM** or **FI** in the ASSA column.

In the vendor master extract, the field denoting the invoice type is determined based on business rules and can typically be set by:

1. The vendor industry key (**NO-PO** invoices typically come from utility/car hire/hotel vendors)
2. Whether the vendor has a purchasing view created
3. The vendor account group
4. The vendor classification

It is also possible to restrict the vendor invoice type determination on a company code-by-company code basis via the following parameter:

```
ITY_VL_SetByVendorCCExceptions=1000,2000
```

If the **SetByVendor** parameter is set to **YES**, the system will not apply the vendor invoice type preference if the invoice is for company codes 1000 or 2000. If the **SetByVendor** parameter is set to **NO**, then the system will only apply the vendor invoice type preference if the invoice is for company codes 1000 or 2000.

Business Requirement 3

The invoice type is determined at the point of scan and passed through the document filename.

```
ITY_VL_Default=PO
ITY_OP_SetByVendor=NO
ITY_VL_SetByVendorCCExceptions=
ITY_VL_POValue=MM
ITY_VL_NPOValue=FI
ITY_OP_SetToPOIfPOFound=NO
ITY_OP_SetToPOIfValidPOFound=NO
ITY_OP_SetToPOIfPOPopulated=NO

IMP_VL_InvoiceType=COMPONENTX
```

In the IMP setting above, *X* represents the underscore-separated component of the document filename where the invoice type is passed from the scanning step. The value of this component for both **PO** and **NO-PO** invoices must be mapped to the **ITY_VL_POValue** and **ITY_VL_NPOValue**. In the example above, the system is expecting to find either **MM** or **FI** in the document filename.

Appendix D Configuring the Vendor ID Field

This section describes how the Vendor ID field can be configured within the AP Packaged Project. The system determines the vendor ID through use of the Associative Search engine, which mandates that each vendor at a single address must have a unique identifier. This unique identifier can be either numeric or alphanumeric.

To ensure compatibility with downstream ERP systems where a single vendor at a single address is denoted by both a vendor ID and a site ID in tandem, or where the ERP system utilizes an external and an internal vendor ID field, the following configuration options are available.

D1. Configuring a Standard Vendor ID

This is the most basic configuration option to implement and should be used in cases where the ERP system provides a single field identifier for a single vendor at a single address.

In this scenario, each record in the vendor extract supplied by the customer, whether it is provided as a CSV file or within a database table or view, should represent a single vendor at a single address and one column in that record should be a unique identifier. The generation of the vendor pool based upon the vendor extract will fail if more than one record shares the same unique identifier.

The following example shows how the vendor configuration can be completed. In this example, the vendor extract has been provided as a CSV file called **vendor.csv**. The steps are as follows:

1. In the project file directory, create a subdirectory called **Pool**.
2. Open the project configuration and configure the *ASA section* to point to the vendor extract file and the newly created pool directory. The following settings are relevant for this:

   ```
   ASA_VL_01_Class=Invoices
   ASA_VL_01_Fieldname=VendorASSA
   ASA_OP_01_AlphaNum=NO
   ASA_OP_01_PoolRelative=NO
   ASA_VL_01_PoolPath=\\NetworkShare\Projects\AP\Global\Pool
   ASA_VL_01_PoolDirectory=Pool
   ASA_VL_01_PoolName=Vendor
   ASA_OP_01_FileRelative=NO
   ASA_VL_01_ImportPathFilename=\\NetworkShare\Projects\AP\Global\vendor.csv
   ASA_VL_01_ImportFilename=vendor.csv
   ```

   If the unique identifier for each row in the vendor extract is numeric, then the *AlphaNum* parameter should be set to **NO**. In all other cases, it should be set to **YES**.

   If Supervised Learning is to be used for the implementation, the path to the pool directory must be populated with a UNC path, and the *PoolRelative* parameter must be set to **NO**. If Supervised Learning is not to be used and the pool directory is located in the same directory as the project file, then the *PoolPath* parameter can be left blank, but the *PoolRelative* parameter must be set to **YES** and the *PoolDirectory* parameter must be populated with the name of the pool directory.

   If UNC paths are used, the relevant directories should have the appropriate shares (usually full control) so that the system can perform the read/write operations required.

   If the vendor extract file is not to be placed in the same directory as the project file, then the *FileRelative* parameter must be set to **NO**, and the *ImportPathFileName* parameter must

be populated with the UNC path to the vendor extract file. If the vendor extract file is located in the same directory as the project file, the *FileRelative* parameter can be set to **YES** and the *ImportFileName* parameter must be populated with the name of the vendor extract file (including the file extension).

3. Open the project file in the Designer application and switch to Definition Mode.
4. Navigate to the **VendorASSA** field on the *Invoices* class, and then show the field's properties.

   The names of the columns shown on the field display are system-assigned names and these names may not actually be indicative of the contents of the field in the vendor extract.
5. Select the **Import** sub-tab of the **Analysis** tab and click the **Import** button to import the pool
6. Select the **General** sub-tab and configure the search fields to be used in identifying the vendor by placing a cross in the respective boxes in the **Search** column. These fields are typically the columns that represent the vendor name, street address, city, zip/postal code and, if appropriate, the vendor telephone numbers and tax/VAT identifiers.
7. Set the radio button in the **ID** column to select the column in the vendor extract that denotes the unique identifier for the vendor record.
8. Go back to the **Import** sub-tab and re-import the pool.
9. Go back to the **General** sub-tab again and configure the **Class Name Format** field. This should always be set to:

   [*unique identifier*]_[*vendor name*]

   If the column called **VENDOR_ID** contains the unique identifier for the vendor and a column called **VENDOR_NAME** contains the vendor name (which may vary on the column names defined in the vendor extract provided by the customer), then the correct entry in this field would be:

   [VENDOR_ID]_[VENDOR_NAME]

   This setting defines how the system will name classes that are built using the Supervised Learning Workflow, but this field must be populated, irrespective of whether the Supervised Learning is being deployed for the implementation or not.
10. Configure the **Field Contents Format**. This defines how the vendor address is displayed on the Verifier form. It is a multi-line field and the first line must always be set to the unique identifier for the record in the vendor extract.

    It is recommended that this setting uses the following structure, but this is optional depending on what is appropriate for the customer.

    [*unique identifier*]
    [*vendor name*]
    [*street address*]
    [*city*]
    [*state or county*] [*zip or postal code*]

    In an example where a column called **VENDOR_ID** represents the unique identifier, the configuration should be as follows:

    [VENDOR_ID]
    [VENDOR_NAME]

[ADDRESS_LINE1]
[ADDRESS_LINE2]
[CITY]
[STATE] [POSTAL_CODE]

11. The *VendorASSA* field configuration is now complete, and a green indicator with the message **Engine is ready!** should appear in the field status box on the **General** sub-tab.
12. Save and close the project file.
13. The final step is to map the identification of each field in the vendor master extract to the relevant fields in the *SRC section* of the project configuration. The system uses this mapping internally for the purposes of displaying the vendor search box and also for applying vendor specific business rules.

    A sample mapping for the example above would be as follows:

```
SRC_VL_ID=VENDOR_ID
SRC_VL_SiteID=
SRC_VL_Name=VENDOR_NAME
SRC_VL_Address1=ADDRESS_LINE1
SRC_VL_Address2=ADDRESS_LINE2
SRC_VL_City=CITY
SRC_VL_Zip=STATE
SRC_VL_State=POSTAL_CODE
SRC_VL_Country=COUNTRY
```

    It is recommended that as many fields in the vendor extract are mapped as possible, which must include the **ID** parameter, along with all the fields shown in the vendor address display box and the vendor's country of origin.

Note: The names of the mapped fields are case sensitive.

D2. Configuring a Vendor and Site ID

These configuration steps should be followed if a single vendor at a single address is represented in the downstream ERP system by a combination of a vendor ID and a site ID. The AP Packaged Project permits use of a composite key within the vendor extract, so the two values will need to be brought together within a single column.

The rule for bringing these two columns together depends on whether the vendor ID and site ID fields are numeric only or alphanumeric.

If the both fields are numeric, the vendor extract will need to contain an additional column representing the vendor ID and site ID combined. The formula that should be followed is:

Unique Identifier = (*Vendor ID* * **1,000,000**) + *Site ID*

For example, if the vendor ID is **1234** and the site ID is **5678** then the unique identifier should be:

(**1234** * **1000000**) + **5678** = **1234005678**

If either the vendor ID or the site ID contains alpha characters, then the formula for combining the two should be:

Unique Identifier = *Vendor ID* & *separator character* & *Site ID*

For example: if the vendor ID is **A12345**, the Site ID is **1000**, and the designated separator is a tilde (~), then the vendor extract should have **A12345~100'** in the unique identifier column.

The nominated separator is specified under the *AlphNumSiteSeparator* parameter in the *VND section* of the project configuration. This must be populated and adhered to if alphanumeric vendor and site IDs are to be used. An error will occur if this has not been configured correctly, or more than one separator is found as part of a single unique identifier.

For ERP systems where a combination of the vendor ID and the site ID represents a single vendor at a single address, the extract file must include three columns; one representing the vendor ID, one representing the site ID, and one representing the combined unique identifier.

The following example shows how the vendor configuration can be completed. In this example, the vendor extract has been provided as a CSV file called **vendor.csv**. The steps are as follows:

1. In the project file directory, create a subdirectory called **Pool**.
2. Open the project configuration and configure the *ASA section* to point to the vendor extract file and the newly created pool directory. The following settings are relevant for this:

   ```
   ASA_VL_01_Class=Invoices
   ASA_VL_01_Fieldname=VendorASSA
   ASA_OP_01_AlphaNum=NO
   ASA_OP_01_PoolRelative=NO
   ASA_VL_01_PoolPath=\\NetworkShare\Projects\AP\Global\Pool
   ASA_VL_01_PoolDirectory=Pool
   ASA_VL_01_PoolName=Vendor
   ASA_OP_01_FileRelative=NO
   ASA_VL_01_ImportPathFilename=\\NetworkShare\Projects\AP\Global\vendor.csv
   ASA_VL_01_ImportFilename=vendor.csv
   ```

   If the unique identifier for each row in the vendor extract is numeric, then the *AlphaNum* parameter should be set to **NO**. In all other cases, it should be set to **YES**.

   If Supervised Learning is to be used for the implementation, the path to the pool directory must be populated with a UNC path, and the *PoolRelative* parameter must be set to **NO**. If Supervised Learning is not to be used and the pool directory is located in the same directory as the project file, then the *PoolPath* parameter can be left blank, but the *PoolRelative* parameter must be set to **YES** and the *PoolDirectory* parameter must be populated with the name of the pool directory.

   If UNC paths are used, the relevant directories should have the appropriate shares (usually full control) so that the system can perform the read/write operations required.

   If the vendor extract file is not to be placed in the same directory as the project file, then the *FileRelative* parameter must be set to **NO**, and the *ImportPathFileName* parameter must be populated with the UNC path to the vendor extract file. If the vendor extract file is located in the same directory as the project file, the *FileRelative* parameter can be set to **YES** and the *ImportFileName* parameter must be populated with the name of the vendor extract file (including the file extension).
3. Open the project file in the Designer application and switch to Definition Mode.
4. Navigate to the **VendorASSA** field on the *Invoices* class, and then show the field's properties.

   The names of the columns shown on the field display are system-assigned names and these names may not actually be indicative of the contents of the field in the vendor extract.
5. Select the **Import** sub-tab of the **Analysis** tab and click the **Import** button to import the pool

6. Select the **General** sub-tab and configure the search fields to be used in identifying the vendor by placing a cross in the respective boxes in the **Search** column. These fields are typically the columns that represent the vendor name, street address, city, zip/postal code and, if appropriate, the vendor telephone numbers and tax/VAT identifiers.
7. Set the radio button in the **ID** column to select the column in the vendor extract that denotes the unique identifier for the vendor record.
8. Go back to the **Import** sub-tab and re-import the pool.
9. Go back to the **General** sub-tab again and configure the **Class Name Format** field. This should always be set to:

   [*vendor id*]_[*vendor name*]

   If the column called **VENDOR_ID** contains the ID for the vendor and a column called **VENDOR_NAME** contains the vendor name (which may vary on the column names defined in the vendor extract provided by the customer), then the correct entry in this field would be:

   [VENDOR_ID]_[VENDOR_NAME]

   This setting defines how the system will name classes that are built using the Supervised Learning Workflow, but this field must be populated, irrespective of whether the Supervised Learning is being deployed for the implementation or not.
10. Configure the **Field Contents Format**. This defines how the vendor address is displayed on the Verifier form. It is a multi-line field and the first line must always be set to the unique identifier for the record in the vendor extract.

    It is recommended that this setting uses the following structure, but this is optional depending on what is appropriate for the customer.

    [*unique identifier*]
    [*vendor name*]
    [*street address*]
    [*city*]
    [*state or county*] [*zip or postal code*]

    In an example where a column called **VENDOR_INDEX** represents the unique identifier, the configuration should be as follows:

    [VENDOR_INDEX]
    [VENDOR_NAME]
    [ADDRESS_LINE1]
    [ADDRESS_LINE2]
    [CITY]
    [STATE] [POSTAL_CODE]
11. The *VendorASSA* field configuration is now complete, and a green indicator with the message **Engine is ready!** should appear in the field status box on the **General** sub-tab.
12. Save and close the project file.
13. The final step is to map the identification of each field in the vendor master extract to the relevant fields in the *SRC section* of the project configuration. The system uses this mapping internally for the purposes of displaying the vendor search box and also for applying vendor specific business rules.

    A sample mapping for the example above would be as follows:

```
SRC_VL_ID=VENDOR_INDEX
SRC_VL_SiteID=VENDOR_SITE_ID
SRC_VL_Name=VENDOR_NAME
SRC_VL_Address1=ADDRESS_LINE1
SRC_VL_Address2=ADDRESS_LINE2
SRC_VL_City=CITY
SRC_VL_Zip=STATE
SRC_VL_State=POSTAL_CODE
SRC_VL_Country=COUNTRY
…
SRC_VL_ExternalVendorID=VENDOR_ID
```

It is recommended that as many fields in the vendor extract are mapped as possible, which must include the **ID** parameter, along with all the fields shown in the vendor address display box and the vendor's country of origin.

The unique identifier should be mapped to the **ID** parameter, the site ID should be mapped to the **SiteID** parameter and the vendor ID should be mapped to the **ExternalVendorID** parameter.

Note: The names of the mapped fields are case sensitive.

D3. Configuring an External Vendor ID

These configuration steps should be followed if the downstream ERP system differentiates between an internal and an external vendor ID, that is to say that it will use an internal vendor ID at the database table level, but the user is presented with an external ID via the application itself. An example of this would be with Oracle Financials, which has a vendor ID field at the database table level, but displays a different supplier number to the user.

If the implementation requires that the Verifier application follows this pattern and displays the external vendor ID to the user, then the vendor extract will require the external vendor ID included as a column, but the unique identifier should always be the ERP system internal vendor ID.

The following example shows how the vendor configuration can be completed. In this example, the vendor extract has been provided as a CSV file called **vendor.csv**. The steps are as follows:

1. In the project file directory, create a subdirectory called **Pool**.
2. Open the project configuration and configure the *ASA section* to point to the vendor extract file and the newly created pool directory. The following settings are relevant for this:

   ```
   ASA_VL_01_Class=Invoices
   ASA_VL_01_Fieldname=VendorASSA
   ASA_OP_01_AlphaNum=NO
   ASA_OP_01_PoolRelative=NO
   ASA_VL_01_PoolPath=\\NetworkShare\Projects\AP\Global\Pool
   ASA_VL_01_PoolDirectory=Pool
   ASA_VL_01_PoolName=Vendor
   ASA_OP_01_FileRelative=NO
   ASA_VL_01_ImportPathFilename=\\NetworkShare\Projects\AP\Global\vendor.csv
   ASA_VL_01_ImportFilename=vendor.csv
   ```

   If the unique identifier for each row in the vendor extract is numeric, then the *AlphaNum* parameter should be set to **NO**. In all other cases, it should be set to **YES**.

If Supervised Learning is to be used for the implementation, the path to the pool directory must be populated with a UNC path, and the *PoolRelative* parameter must be set to **NO**. If Supervised Learning is not to be used and the pool directory is located in the same directory as the project file, then the *PoolPath* parameter can be left blank, but the *PoolRelative* parameter must be set to **YES** and the *PoolDirectory* parameter must be populated with the name of the pool directory.

If UNC paths are used, the relevant directories should have the appropriate shares (usually full control) so that the system can perform the read/write operations required.

If the vendor extract file is not to be placed in the same directory as the project file, then the *FileRelative* parameter must be set to **NO**, and the *ImportPathFileName* parameter must be populated with the UNC path to the vendor extract file. If the vendor extract file is located in the same directory as the project file, the *FileRelative* parameter can be set to **YES** and the *ImportFileName* parameter must be populated with the name of the vendor extract file (including the file extension).

3. Open the project file in the Designer application and switch to Definition Mode.
4. Navigate to the **VendorASSA** field on the *Invoices* class, and then show the field's properties.

   The names of the columns shown on the field display are system-assigned names and these names may not actually be indicative of the contents of the field in the vendor extract.
5. Select the **Import** sub-tab of the **Analysis** tab and click the **Import** button to import the pool
6. Select the **General** sub-tab and configure the search fields to be used in identifying the vendor by placing a cross in the respective boxes in the **Search** column. These fields are typically the columns that represent the vendor name, street address, city, zip/postal code and, if appropriate, the vendor telephone numbers and tax/VAT identifiers.
7. Set the radio button in the **ID** column to select the column in the vendor extract that denotes the unique identifier for the vendor record.
8. Go back to the **Import** sub-tab and re-import the pool.
9. Go back to the **General** sub-tab again and configure the **Class Name Format** field. This should always be set to:

   [*vendor number*]_[*vendor name*]

   If the column called **VENDOR_NUMBER** contains the external ID for the vendor and a column called **VENDOR_NAME** contains the vendor name (which may vary on the column names defined in the vendor extract provided by the customer), then the correct entry in this field would be:

   [VENDOR_NUMBER]_[VENDOR_NAME]

   This setting defines how the system will name classes that are built using the Supervised Learning Workflow, but this field must be populated, irrespective of whether the Supervised Learning is being deployed for the implementation or not.
10. Configure the **Field Contents Format**. This defines how the vendor address is displayed on the Verifier form. It is a multi-line field and the first line must always be set to the unique identifier for the record in the vendor extract.

It is recommended that this setting uses the following structure, but this is optional depending on what is appropriate for the customer.

[*unique identifier*]
[*vendor name*]
[*street address*]
[*city*]
[*state or county*] [*zip or postal code*]

In an example where a column called **VENDOR_INDEX** represents the unique identifier, the configuration should be as follows:

[VENDOR_INDEX]
[VENDOR_NAME]
[ADDRESS_LINE1]
[ADDRESS_LINE2]
[CITY]
[STATE] [POSTAL_CODE]

11. The *VendorASSA* field configuration is now complete, and a green indicator with the message **Engine is ready!** should appear in the field status box on the **General** sub-tab.
12. Save and close the project file.
13. The final step is to map the identification of each field in the vendor master extract to the relevant fields in the *SRC section* of the project configuration. The system uses this mapping internally for the purposes of displaying the vendor search box and also for applying vendor specific business rules.

    A sample mapping for the example above would be as follows:

```
SRC_VL_ID=VENDOR_INDEX
SRC_VL_SiteID=VENDOR_SITE_ID
SRC_VL_Name=VENDOR_NAME
SRC_VL_Address1=ADDRESS_LINE1
SRC_VL_Address2=ADDRESS_LINE2
SRC_VL_City=CITY
SRC_VL_Zip=STATE
SRC_VL_State=POSTAL_CODE
SRC_VL_Country=COUNTRY
…
SRC_VL_ExternalVendorID=VENDOR_NUMBER
```

    It is recommended that as many fields in the vendor extract are mapped as possible, which must include the **ID** parameter, along with all the fields shown in the vendor address display box and the vendor's country of origin.

    The unique identifier should be mapped to the **ID** parameter, the site ID should be mapped to the **SiteID** parameter and the external vendor ID should be mapped to the **ExternalVendorID** parameter.

Note: The names of the mapped fields are case sensitive.

D4. Working With Addresses in Non-Western Languages

If the vendor extract contains non-Western characters the **Brainware V3** Associative Search Engine must be selected. In the project OCR settings for FineReader the appropriate OCR languages for the documents being processed must also be selected.

Note: Version 3 of the Associate Search Engine is only available in WebCenter Forms Recognition version 11.1.1.8.0 or later.

If the Verifier client application is being used, the desktop locale must be set to use the non-Western language as the language for non-unicode applications. This carries the limitation that only one non-Western alphabet can be displayed in the Verifier application at any one time.

Multiple non-Western language display is supported within Web Verifier deployments.

Appendix E Configuring Miscellaneous Charges

E1. Introduction

Miscellaneous charges refer to the additional items a vendor may include on an invoice document that have to be booked with the primary invoice items.

Examples include a freight charge, a customs charge, an energy surcharge or an administration code.

Miscellaneous charges can appear at both the header and item level on an invoice.

At the header level, the AP Packaged Project provides two fields for this purpose:

- AmountFreightPrepaidAndAdded
- AmountMisc.

At the line item level, the *Category* column is used to identify and classify any captured miscellaneous charges that the vendor specifies as a line item on the invoice. This category is set automatically by the project, drawing on the configuration settings applied in the *MSC section*.

The handling of miscellaneous charges is dependent on the business rules of the customer. The configuration options within the *MSC section* permit:

- Miscellaneous charges to be booked as unplanned delivery costs (i.e. a single, header-level amount field within the downstream ERP system).
- Miscellaneous charges to be booked as a separate invoice line item with a specific line type (Oracle Financials).
- Miscellaneous charges to be booked as a direct general ledger account entry.

These events occur during the line pairing operation carried out during document export. Line pairing must be activated in the *LPR section* of the project configuration for miscellaneous charge processing to occur. If line pairing is deactivated, then any miscellaneous charges that appear on the invoice will be outputted in the form in which they were captured.

Note: The system will always check for the existence of a regular purchase order line item representing the miscellaneous charge before applying the configured processing option. If one is found, then this line item will be used. Miscellaneous charges set up as regular purchase order line items are identified by the purchase order line item description and the extent to which it fits with the aliases configured.

E2. Miscellaneous Charge Categories

Each group within the *MSC section* of the project configuration represents a miscellaneous charge category. The category denotes the type of miscellaneous charge (e.g. freight, a customs charge, etc.).

The number of categories that should be configured is dependent on how the customer wishes to process these charges, and the number of miscellaneous charge categories is driven by the number of different ways they wish to do it.

For example, if all miscellaneous charges, irrespective of what they are, should be summed and booked as a single general ledger account entry in the downstream ERP system, then only one miscellaneous charge category needs to be configured.

If, however, the customer wishes to book specific miscellaneous charges to a specific general ledger account depending on what type of charge it was, then there would need to be as many miscellaneous charge categories as there are possible general ledger codes to book them against.

Within each group, a code and name is assigned to each miscellaneous charge category. In the Verifier application, if a line item is identified as belonging to a particular miscellaneous charge category, the code set for that group in the project configuration is copied into the *Category* column for that particular line item.

E3. Assigning Header Fields to a Miscellaneous Charge Category

Each miscellaneous charge category can be assigned fields at either the header or line item level.

In the *MSC section* of the project configuration, the parameter **NN_HeaderField** controls the mapping between a project header field and the miscellaneous charge category.

The two standard header fields available for mapping are *AmountFreightPrepaidAndAdded* and *AmountMisc*. A sample configuration for mapping the Freight field to the *FREIGHT* category is below:

```
MSC_VL_01_Type=FREIGHT
MSC_VL_01_Code=F
MSC_VL_01_HeaderField=AmountFreightPrepaidAndAdded
```

More than one header field can be mapped to a single miscellaneous charge category via comma-separation:

```
MSC_VL_01_HeaderField=AmountFreightPrepaidAndAdded,AmountMisc
```

Note: The name of the project header field is not case-sensitive.

E4. Assigning Line Items to a Miscellaneous Charge Category

The assignment of line items to a miscellaneous charge occurs via the extracted line item description.

In the **NN_Alias** parameter, a comma-separated list of keywords should be entered to indicate that the line item belongs to the miscellaneous charge group, for example:

```
MSC_VL_01_Type=FREIGHT
MSC_VL_01_Code=F
MSC_VL_01_HeaderField=AmountFreightPrepaidAndAdded
MSC_VL_01_Alias=FREIGHT,DELIVERY,CARRIAGE,UPS,TRANSPORT
```

Note: The list of miscellaneous charge aliases is not case-sensitive.

E5. Posting Miscellaneous Charges as a Specific Line Type

If the business rule for processing charges belonging to the miscellaneous charge category involves booking them as a specific line type in the downstream ERP (e.g. Oracle Financials), then

the required line type should be entered in the **NN_LineType** parameter. For example, if the ERP line type for freight is **FRT** the following should be populated:

```
MSC_VL_01_Type=FREIGHT
MSC_VL_01_Code=F
MSC_VL_01_HeaderField=AmountFreightPrepaidAndAdded
MSC_VL_01_Alias=FREIGHT,DELIVERY,CARRIAGE,UPS,TRANSPORT
MSC_VL_01_LineType=FRT
```

If the line type value is populated, this will over-ride any further handling settings applied to the miscellaneous charge group.

At point of export, the miscellaneous charge will be outputted as a line item.

E6. Posting Miscellaneous Charges as Unplanned Costs

If the business rule for processing charges belonging to the miscellaneous charge category involves always booking them as an unplanned cost in the downstream ERP system, then this is configured via the **NN_AlwaysBookedToUnplanned** parameter. For example:

```
MSC_VL_01_Type=FREIGHT
MSC_VL_01_Code=F
MSC_VL_01_HeaderField=AmountFreightPrepaidAndAdded
MSC_VL_01_Alias=FREIGHT,DELIVERY,CARRIAGE,UPS,TRANSPORT
MSC_VL_01_LineType=
MSC_OP_01_AlwaysBookToUnplanned=YES
```

At time of export, the total value of all miscellaneous charges in every category that require handling as unplanned delivery costs will be written to the **EXP_VL_DBHCUnplannedFreight** field for database output.

E7. Posting Miscellaneous Charges to a General Ledger Account

If this option is selected, the system will create a separate general ledger entry for the miscellaneous charge. A general ledger entry consists of the following components:

- A general ledger account code representing the type of expense.
- Cost object (e.g. a cost center, an internal order, a profit center, a project etc.)
- A tax code.

To determine the general ledger account code, the system will firstly look into a custom database table. Executing the **WFR_AP_ Tables_Create_<***database***>.sql** script provided with the AP Packaged Project will create this table with the name **OFRMISCACC**. The customer is responsible for populating this table with the appropriate data.

If an entry exists for the invoice company code and the miscellaneous charge code, then this is the general ledger account code that the system will use. If no entry exists, or the custom table lookup has not been activated via the *MSC_OP_ValidateFromDB* parameter, then the default general ledger code for the miscellaneous charge group will be used.

The sequential derivation of the cost object is as follows:

1. If the *NN_GetCostObjectFromPO* parameter against the miscellaneous charge group is set to **YES**, the cost object will be retrieved from the first invoice line paired that uses either a cost center/profit center, an internal order or a project.

2. If a profit center or cost center exists in the custom table, this will be used as the cost object.
3. The default cost center or profit center, in *NN_DefaultCostCenter* or *NN_DefaultProfitCenter* respectively, is used.

The sequential derivation of the tax code is as follows:

1. If an entry exists in the custom table for the invoice company code and miscellaneous charge category, this tax code is used.
2. The tax code is retrieved from the first paired invoice line item.
3. The default tax code set in the *NN_DefaultTaxCode* parameter is used.

For countries and ERP systems that use tax jurisdictions, the tax jurisdiction code is derived from the first paired invoice line.

If the general ledger account is not relevant for tax postings in the ERP system, a double asterisk (**) should be entered into the tax code column in the table. This will have the effect of passing a blank tax code and a blank tax jurisdiction code downstream.

If the custom table lookup is activated, but communication does not succeed either due to missing or incorrect configuration, or database unavailability, the export event for the document will fail.

A sample configuration for posting miscellaneous charges as general ledger entries is shown below:

```
MSC_VL_01_Type=FREIGHT
MSC_VL_01_Code=F
MSC_VL_01_HeaderField=AmountFreightPrepaidAndAdded
MSC_VL_01_Alias=FREIGHT,DELIVERY,CARRIAGE,UPS,TRANSPORT
MSC_VL_01_LineType=
MSC_OP_01_AlwaysBookToUnplanned=NO
MSC_OP_01_AlwaysBookToPlanned=NO
MSC_VL_01_ValidConditions=
MSC_OP_01_AlwaysBookToGLAccount=YES
MSC_OP_01_BookToUnplannedIfNoPlanned=NO
MSC_OP_01_BookToGLAccountIfNoPlanned=NO
MSC_VL_01_GLAccount=1000
MSC_OP_01_GetCostObjectFromPOLine=YES
MSC_VL_01_DefaultCostCenter=1000
MSC_VL_01_DefaultProfitCenter=2000
MSC_VL_01_DefaultTaxCode=I0
```

With the configuration above, if the custom table is not used, the system will book the miscellaneous charge as a general ledger entry using general ledger account **1000** with the cost object and tax code from the first paired invoice line. If no cost object exists against the first paired invoice line, then cost center **1000** and profit center **2000** will be used. If no tax code exist against the first paired invoice line, tax code **I0** will be used.

E8. General Ledger Account Code Table

To facilitate greater versatility when posting freight as general ledger account entries, a lookup table is available so that specific general ledger account codes, cost objects and tax codes can be assigned on a company code by company code basis and a plant by plant basis incorporating the purchase order line type if required.

Executing the **WFR_AP_ Tables_Create_<*database*>.sql** script provided with the AP Packaged Project will create this table with the name **OFRMISCACC**. The customer is responsible for populating this table with the appropriate data. An example for populating the table for company code **GB01**, plant **1000**, line type **A** and miscellaneous charge category **F** is shown below:

COMPANY CODE	CATEGORY	LINE TYPE	PLANT	GL ACCOUNT	COST CENTER	PROFIT CENTER	TAX CODE
GB01	F	A	1000	1000	2000	3000	I0

The company code, plant, line type and category columns are mandatory and form the unique key for each record in the table.

If the general ledger account, cost object and tax codes are to be set at a company code level rather than a plant level, a space should be entered into the plant column. If no line type is to be used, a space should also be entered into the line type column.

The system will access the table according to the following sequence:

1. By company code, plant and line type.
2. By company code and line type.
3. By company code and plant.
4. By company code alone.

As soon as a matching record is found, the access sequence will break and that record will be used. The above sequence should be considered when populating the table.

E9. Third Party Freight

Within the AP Packaged Project, a third party freight invoice refers to a very specific business scenario whereby an invoice is received from a vendor billing for freight, yet that vendor legitimately quotes the purchase order number of another vendor (the material vendor), and it's against this material vendor's purchase order that the freight charge needs to be booked.

Note: Freight invoices that do not fall into this category are handled as regular invoices.

During line pairing, the system will book the net value of the invoice against the material vendor's purchase order, according to the rules set against the miscellaneous charge group assigned to third party freight vendors. For example:

```
MSC_VL_ThirdPartyFreightCode=F
```

This denotes that the rules should be derived from whichever miscellaneous charge group has been assigned a code of **F**.

If the rule is set to post as unplanned, the system will also create a zero-value subsequent debit invoice line against the first line item of the purchase order belonging to the material vendor.

If the rule is set to post against planned condition types on the purchase order, these condition types must be goods receipted.

E10. Adding New Miscellaneous Charge Fields

The AP Packaged Project comes with two fields defined for capturing miscellaneous charges at the header level, namely *AmountFreightPrepaidAndAdded* and *AmountMisc*.

The *AmountFreightPrepaidAndAdded* field is pre-trained to extract freight and transportation costs from invoices. The *AmountMisc* field is not trained so that flexibility is there to designate that field for a specific type of miscellaneous charge, for example a pallet charge or an energy surcharge. The *UserExitAmountMiscPostEvaluate* script procedure is available for any custom script deemed appropriate to assist with the extraction of this field.

Should a third field be required, this should be created at the *Invoices* class level, and should be named **AmountMisc***XXX* where *XXX* is a meaningful field descriptor. This will ensure that the new miscellaneous charge field is included in the mathematical validation applied to the invoice header-level amounts.

Additionally, a standard validation event for the new field should be added to the *Invoices* class level, containing script to call the system header level validation function *fnValidateAmount*. This function will also handle the field formatting.

For example, assume a new header-level field is added to the *Invoices* class and is named **AmountMiscPalletCharge**. The following script should be added to the *Validate* event of this new field:

```
Private Sub AmountMiscPalletCharge_Validate(pField As
SCBCdrPROJLib.ISCBCdrField, pWorkdoc As SCBCdrPROJLib.ISCBCdrWorkdoc,
pValid As Boolean)

    pValid = fnValidateAmount(pField, pWorkdoc)

End Sub
```

Should a *PostEvaluate* event be required to assist with the extraction of the custom miscellaneous charge field, this should also be created on the *Invoices* class level.

Appendix F Deactivating ASSA Fields

This section describes the two aspects to deactivating ASSA fields within the AP Packaged Project.

F1. Deleting the ASSA Entries from the Project Configuration File

Note: Steps 1 to 5 below are only required for AP Packaged Project versions prior to 1004. For later versions of the project, continue to *Section F2: Deactivating the Associative Search Engine*.

1. Open the project configuration file in a text editor.
2. Navigate to the **ASA** section.
3. Find the field that you wish to deactivate and its associated class.

   The standard delivered fields are:

   - VendorASSA
   - CompanyCode *(already deactivated in the project file for AP Packaged Project 1007G)*
   - EmployeeASSA *(already deactivated in the project file for AP Packaged Project 1007G)*
4. Either delete the settings group for the field or comment each line for the group out with either a single quote or hash character as the first character of the line.
5. Save the changes and close the text editor.

F2. Deactivating the Associative Search Engine

1. Open the project file using the WebCenter Forms Recognition Designer application, and login as an administrative user.
2. Switch to Definition Mode.
3. Navigate to the class that holds the definition of the field, then to the field itself.
4. Show the field's properties.
5. Select **No Analysis Engine** from the *Available Analysis Engines* dropdown.
6. Save the project and close the Designer application.

Appendix G Handling Intercompany Vendors

G1. Introduction

Intercompany invoices arise in instances where different subsidiaries of the same group of companies are invoicing one another. For example, where *ACME Industries France* sends an invoice to *ACME Industries UK*. This phenomenon is especially prevalent in large multi-national corporations where a substantial proportion of the daily invoice volume can represent this kind of transaction.

Intercompany invoices pose a particular challenge for the AP Packaged Project in the sense that they need to be handled separately from regular third party invoices. The reason for this is that the presence of bill-to address information on a third party invoice coupled with the presence of intercompany vendor addresses in the vendor extract can skew the recognition rates of third party vendors. Hence, the system could potentially propose an intercompany vendor as the best vendor for a third party invoice by virtue of the fact that the intercompany vendor details represent a good fit to the bill-to address details provided on the invoice.

Hence, for implementations that are required to handle intercompany invoices, the AP Packaged Project provides a dedicated *Intercompany* class for this purpose.

G2. Vendor Extracts

If intercompany invoices are in scope for the solution deployment, then it is incumbent on the customer to provide two vendor extract files: one which exclusively contains third party vendors, and another which exclusively contains intercompany vendors.

The third party vendor extract file is set up in the normal manner. The intercompany vendor extract file is set up in the same way except that it is mapped to the *VendorASSA* field at the *Intercompany* class level.

G3. Configuring the Intercompany Classification

Once the vendor extract pools have been set up, the next step is to ensure that the intercompany invoices are directed to the **Intercompany** class.

This is not a mandatory activity, but the consequence of not doing so will be that all intercompany invoices will be classified to the *Invoices* class, and will stop in Verifier on the grounds that the system was unable to find the correct vendor. The user would, therefore, have to reclassify each intercompany invoice manually.

This may be acceptable if the volume of intercompany invoices is particularly low, otherwise a strategy for correct automatic classification is required.

Selecting an appropriate classification strategy is dependent on a number of factors, which include the following:

1. The volume of intercompany invoices.
2. The variety of intercompany invoice formats.
3. The extraction success on intercompany invoices through the generic learnset.
4. The degree of document separation at time of scanning.

The following examples provide some direction upon the best approaches for classification depending on client circumstances.

Example 1: Intercompany invoices are separated from third party vendor invoices at the point of scan.

If intercompany invoices are separated from third party vendor invoices, and are scanned using a different scan job, then this can be used as a means to classify them to the correct class.

A simple example of this would be if the intercompany scan job outputs a TIFF image with an element built into the filename that denotes that it is an intercompany document, for example, the first two characters of the TIFF filename are **IC**.

In this case, a simple script in the *UserExitPostClassify* procedure can be used to force the document to the **Intercompany** node, thus ensuring correct classification to the extent that the document was scanned correctly in the first place. In the event of a misscan, the document will be classified to the *Invoices* class and a user will have to reclassify it to the *Intercompany* class manually.

A sample script for the above scenario is shown below:

```
Public Sub UserExitPostClassify(pWorkdoc As SCBCdrPROJLib.SCBCdr.Workdoc)

   Dim strFileName As String

   strFileName = fnGetFileName(pWorkdoc.DocFileName(0))

   If Left(strFileName, 2) = "IC" Then
      pWorkdoc.DocClassName = "Intercompany"
   End If

End Sub
```

The function *fnGetFileName* provides a simple means to extract the filename minus the filepath and file extension. The name of the *Intercompany* class is case-sensitive and should always be referred to as **Intercompany**.

Example 2: Client processes a significant volume of intercompany invoices, but there are 5 different formats or fewer.

If the number of intercompany invoice formats total five or less, then an example of each format can be added directly to the classification learnset for the *Intercompany* class using the Template Classification Engine. Further examples of each format can be added until the maximum of five learnset documents is reached.

The five-document limit can theoretically be increased to allow further examples of the same invoice format via the *Max Number of Templates per Class* property of the Template Classification Engine.

It is not recommended to include more than five different invoice formats in a single Template Classification-based learnset.

It is not necessary to train a corresponding extraction learnset for the *Intercompany* class, as the class will automatically inherit the extraction training from the generic learnset held at the *Invoices* class level.

If, however, the generic extraction is not delivering acceptable extraction results, an extraction learnset exclusive to the intercompany invoices may be created at the *Intercompany* level using Normal Train Mode in the Designer application.

It is not possible to use the Supervised Learning Workflow feature in conjunction with the *Intercompany* class. Intercompany classes can only be created or amended through the Designer application.

Example 3: Client processes a significant volume of intercompany invoices, and there is a wide variety of different formats.

Large organizations with many different subsidiaries may experience high volumes of intercompany invoices in a multitude of different invoice formats.

If this is the case, then it is recommended to create new intercompany subclasses under the existing *Intercompany* class to accommodate the different format examples required to build a classification learnset.

Depending on volumes and the success of extraction using the generic learnset, it may also be advantageous to create a dedicated extraction learnset for each format. There is no restriction on the number of subclasses that can be created, although it is recommended that each have a title indicative of the formats of documents it is designed to handle. For large numbers of subclasses, it may be desirable to use the Layout Classification Engine in lieu of the Standard Template Classification engine.

Appendix H Configuring the VAT Number Compliance Check

H1. Introduction

The AP Packaged Project provides a VAT registration number compliance check to satisfy a European Union legal requirement that applies to invoices that include Value Added Tax.

This requirement states that if VAT is to be charged on the document, it is incumbent upon the vendor to quote not only their own VAT registration number (the vendor VAT registration number), but also the VAT registration number of the party being invoiced (the bill-to VAT registration number).

EU cross-border transactions, where the VAT is zero-rated, but VAT registration numbers are still required, are discussed in *Section H4: Cross-Border EU VAT Registration Number Checks*.

Both VAT registration numbers must be valid for the same country, which is why VAT is generally most prevalent on domestic invoices, for example, where a UK vendor is invoicing another UK company, or a German vendor is invoicing another German company.

A German company would not charge VAT on an invoice to a UK company unless one of the following conditions is met:

1. The German company is registered for VAT in the UK.
2. The UK company is registered for VAT in Germany.
3. The UK company is not registered for VAT at all.

In the first instance, the German vendor would be required to specify their UK VAT registration number on the invoice, as well as the UK VAT registration number of their customer. The VAT charged would come under the jurisdiction of the UK tax authorities, so UK VAT rates must be applied. The German vendor would not be allowed to charge tax in line with German VAT rates.

In the second instance, the German vendor would be required to specify their German VAT registration number on the invoice, as well as the German VAT registration number of their customer. The VAT charged would fall under the jurisdiction of the German tax authorities, so the national German rates of VAT would apply.

In the third instance, VAT would be levied as if the UK company were a private individual. Under this circumstance, the German company is correct in charging VAT at the German rate, even if the goods were destined for the UK.

Failure to quote both VAT registration numbers on the document constitute a non-compliant invoice under Article 226 of EU Council Directive 2006/112/EC, and the party being invoiced would be within their rights to reject the invoice back to the vendor. In reality, vendors are seldom inclined to quote the VAT registration number of the customer for domestic transactions.

If the VAT compliance check is activated, the AP Packaged Project will send a document to Verifier if all of the following conditions are met:

1. Tax is being charged on the document.
2. Both the vendor and the company being invoiced are based in an EU member state.

3. One or both of the VAT registration numbers are not present on the document, or have not been captured automatically, or do not share the same country prefix. The system may be configured only to require the VAT registration number of the vendor.

A further requirement is that if VAT is to be charged, the currency of the invoice must be the local currency of the country whose VAT rates are being applied. In the example above, if the German vendor is to issue an invoice charging UK VAT, the invoice currency must be British Pounds. If the invoice is issued in the local currency of Germany then the invoice must quote either of the following:

1. The British Pounds equivalent of the VAT amount.
2. The exchange rate at time of invoice issue between British Pounds and Euros.

The AP Packaged Project will detect the above circumstance and will require the user to enter either a local VAT amount or an exchange rate if no value has been captured automatically. This check is carried out once one or a pair of valid VAT registration numbers has been found. The currency of the company code is used as the currency in which either the invoice as a whole or the local VAT should be quoted.

H2. Configuring the Extraction

To activate the extraction of both the vendor and bill-to VAT registration numbers, the following parameter should be set to **YES** in the *TAX section* of the project configuration:

```
TAX_OP_ActivateVATComplianceCheck=YES
```

This can be done on a company code by company code basis if required. Company codes that are to be excluded should be specified as a comma-separated list against the following parameter:

```
TAX_VL_VATCheckCompanyCodeExceptions=1000,2000,3000
```

In the example above, the VAT registration number extraction will be skipped for all invoices intended for company codes **1000**, **2000** and **3000**. If the *ActivateVATComplianceCheck* parameter is set to **NO**, then the inverse will be true and the VAT registration number extraction will only be carried out for invoices relating to company codes **1000**, **2000** and **3000**.

In order for the system to be able to extract the VAT registration number of the vendor, the VAT registration number must be populated within the vendor extract used by the project, and the column that denotes the VAT registration number must be mapped in the *SRC section* of the project configuration. For example:

```
SRC_VL_VATRegNo=VAT_REGISTRATION_NUM
```

To activate the extraction of the bill-to VAT registration number, the company code lookup must be activated in the *CCO section* of the project configuration.

If the company code lookup is to be against a database then the following parameter must be set to **YES**:

```
CCO_OP_ValidateFromDB=YES
```

The connection group representing the connection string that points to the database containing the company code lookup table in the *SQL section* should be configured against the following parameter:

```
CCO_VL_SQLConnectionGroup=01
```

The final step is to map the technical names of the table and its associated columns to the remaining database lookup parameters.

The table below is an example populated database table containing company code lookup information. Executing the **WFR_AP_ Tables_Create_<*database*>.sql** script provided with the AP Packaged Project will create a table called **OFRCOMPANY** with this structure but it is the responsibility of the customer to populate that table.

COMPANYCODE	COUNTRY	CURRENCY	VATREGNO
GB01	GB	GBP	GB123456789
DE01	DE	EUR	DE123456789

For the table structure above, the parameters should be configured as follows:

```
CCO_VL_DBTableName=OFRCOMPANY
CCO_VL_DBColumnName=COMPANYCODE
CCO_VL_DBCountry=COUNTRY
CCO_VL_DBCurrency=CURRENCY
CCO_VL_DBVATRegNos=VATREGNO
```

Once the above configuration has been completed, and the VAT registration number values are populated in the vendor extract and company code validation tables, the system will attempt to extract them from the document.

The values will not be extracted if any of the following conditions are met:

1. A vendor cannot be determined from the invoice.
2. A company code cannot be determined from the invoice.
3. The company code is listed as an exception company code.
4. The user has selected an invalid reason other than **NONE**, **THIRD PARTY FREIGHT**, **PO VENDOR IS NOT INVOICE VENDOR** or **STOCK INVOICE**.
5. The registration numbers are not present on the document, or the document was of sufficient bad quality so that they could not be read.

Multiple VAT registration numbers, either per-vendor or per-company code, are supported. These should be included in the relevant columns in the vendor extract or the company code table as a comma-separated list.

H3. Configuring the Validation

The final step to completing the compliance check is to configure the validation so the system knows that the vendor and the company code belong to EU member states.

The system expects a flag to indicate whether the vendor belongs to an EU member state to be present in the vendor extract.

In the *SRC section* of the project configuration, the following parameters must be completed so that the system can interpret the flag correctly:

```
SRC_VL_EUMember=ISEUMEMBER
SRC_VL_EUMemberAlias=X
```

The parameter values in the example above are provided as sample entries and may be different for a specific implementation.

The *EUMemberAlias* parameter contains the value that denotes a positive identification of an EU member state. In the example above, if a value of **X** is found in the vendor extract column called **ISEUMEMBER**, the system will interpret that as meaning that the vendor belongs to an EU member state. If any other value is found, the system will conclude that the vendor does not belong to an EU member state.

The next step is to configure the system to recognize that the invoice company code also belongs to an EU member state country. The **WFR_AP_ Tables_Create_<*database*>.sql** script provided with the AP Packaged Project must be executed to create the **OFRCOUNTRY** database table and populate it with all world countries and a flag denoting whether that country is an EU member state.

Once this table is installed, the parameters in the *CTR section* should be completed as follows with the appropriate SQL connection group:

```
CTR_OP_ValidateFromDB=YES
CTR_VL_SQLConnectionGroup=01
CTR_VL_DBTableName=OFRCOOUNTRY
CTR_VL_DBCountry=COUNTRY
CTR_VL_DBCurrency=CURRENCY
CTR_VL_DBEUMember=EUMEMBER
CTR_VL_DBName=COUNTRYNAME
```

The system will now set the vendor and bill-to VAT registration numbers to invalid if:

1. One or both is missing.
2. VAT is being charged.
3. The two-character country prefixes do not match.

If both VAT registration numbers are present and the invoice currency is different to the currency of the company code, the system will prompt the user to enter the VAT amount in the local currency of the VAT registration number prefix country or an exchange rate.

It is also possible to limit the check only to require the VAT registration number of the vendor. To do this, the following parameter should be set to **YES**:

```
TAX_OP_VendorVATCheckOnly=YES
```

The vendor VAT registration number-only check can also be configured so that it only applies to certain company codes. This is controlled via the following parameter:

```
TAX_VL_VendorVATCheckCompanyCodeExceptions=1000,2000,3000
```

Company codes that only require the vendor VAT registration details should be entered against the parameter above as a comma-separated list. In the example above, the vendor-only VAT compliance check will be skipped for all invoices intended for company codes **1000**, **2000** and **3000**. In other words, the bill-to VAT registration number will be required as well. If the *VendorVATCheckOnly* parameter is set to **NO** the inverse is true and the vendor-only VAT compliance check will only be carried out for invoices relating to company codes **1000**, **2000** and **3000**.

H4. Cross-Border EU VAT Registration Number Checks

For cross-border EU transactions, the vendor may zero-rate the VAT charged on their invoice, but, if they do that, they must specify both their VAT registration number and the VAT registration number of their customer on the document.

In this scenario, their customer is obliged to calculate and declare the VAT at their local rate to the local tax authorities as if the purchase had been made from a domestic vendor.

To configure the VAT registration number compliance check for cross-border EU transactions, both the Extraction and the Validation steps detailed in this Appendix must be completed. Additionally, the following parameter must be set to **YES**:

```
TAX_OP_CheckVATCrossBorder=YES
```

The system will now send a document to Verifier if all of the following conditions are met:

1. Zero tax is being charged.
2. Both the vendor and company code are domiciled in different EU countries.
3. The vendor has a VAT registration number populated in the vendor extract.
4. Either the vendor VAT registration number or the bill-to VAT registration number is missing or has not been captured from the document.

This check works independently of the vendor-only VAT registration number check, so both VAT registration numbers will always be required.

It is possible to configure the cross-border VAT registration number check on a company code by company code basis via the following parameter:

```
TAX_VL_CrossBorderCompanyCodeExceptions=1000,2000,3000
```

Company codes that do not require the cross-border VAT check should be entered against the parameter above in the form of a comma-separated list. In the example above, the cross-border VAT compliance check will be skipped for all invoices intended for company codes **1000**, **2000** and **3000**. If the *CheckVATCrossBorder* parameter is set to **NO**, then the cross-border VAT compliance check will only be carried out for invoices relating to company codes **1000**, **2000** and **3000**.

Appendix I Activating Optional Project Fields

I1. Introduction

In the standard version of the AP Packaged Project, the Delivery Note Number, Payment Reference, Due Date and Delivery Date fields are not activated. This is in order to maximize system performance for projects in which these fields are not needed.

However, should one or more of these fields be required in an implementation, they can be activated as described below.

I2. Activating the Payment Reference Field

Perform the following steps to activate the Payment Reference field in the AP Packaged Project:

1. Open the project in the Designer application and login as an administrative user
2. From the **View** menu, select the **Definition Mode** option
3. Double-click the **Invoices** class node in the **Classes** tree to display the fields for the class
4. Select the **[33] PaymentReference** field, then from the **Edit** menu, select the **Show Properties** option to display the properties pane
5. On the **Analysis** tab on the properties pane, select the **PaymentReference** item from the **Available Templates** dropdown
6. Save the project

I3. Activating the Delivery Note Number Field

Perform the following steps to activate the Delivery Note Number field in the AP Packaged Project:

1. Open the project in the Designer application and login as an administrative user
2. From the **View** menu, select the **Definition Mode** option
3. Double-click the **Invoices** class node in the **Classes** tree to display the fields for the class
4. Select the **[56] DeliveryNote** field, then from the **Edit** menu, select the **Show Properties** option to display the properties pane
5. On the **Analysis** tab on the properties pane, select the **DeliveryNote** item from the **Available Templates** dropdown
6. Save the project

I4. Activating the Due Date Field

Perform the following steps to activate the Due Date field in the AP Packaged Project:

1. Open the project in the Designer application and login as an administrative user
2. From the **View** menu, select the **Definition Mode** option
3. Double-click the **Invoices** class node in the **Classes** tree to display the fields for the class

4. Select the **[57] DueDate** field, then from the **Edit** menu, select the **Show Properties** option to display the properties pane
5. On the **Analysis** tab on the properties pane, select the **DueDate** item from the **Available Templates** dropdown
6. Save the project

I5. Activating the Delivery Date Field

Perform the following steps to activate the Delivery Date field in the AP Packaged Project:

1. Open the project in the Designer application and login as an administrative user
2. From the **View** menu, select the **Definition Mode** option
3. Double-click the **Invoices** class node in the **Classes** tree to display the fields for the class
4. Select the **[58] DeliveryDate** field, then from the **Edit** menu, select the **Show Properties** option to display the properties pane
5. On the **Analysis** tab on the properties pane, select the **DeliveryDate** item from the **Available Templates** dropdown
6. Save the project

Appendix J Unit of Measure Conversions

J1. Introduction

Units of measure appear at line item level on an invoice. They describe the magnitude of the physical quantity to which the line item quantity relates. Common units of measure found on material invoices include **each**, **kilograms**, **lbs**, **boxes** and **cases**. For service invoices, that are billing time expended, **hours** and **days** are commonly used.

The AP Packaged Project will look for a unit of measure when extracting the invoice line items and a designated column is included in the line items table for that reason. This is not a mandatory entry.

When exporting the line items, the unit of measure passed downstream for a paired line item will always be the unit of measure that was retrieved from the purchase order line. This is typically mandated by the downstream ERP system. If the invoice and purchase order units of measure differ, the system needs to adjust the quantity so that it is expressed in the purchase order unit of measure to prevent the wrong quantity being booked.

Example 1:

The invoice has a line item for a quantity of **1000 kilograms**, and the corresponding purchase order line item is for **1 tonne**. Therefore, when exporting the line item, a unit of measure of **tonne** will always be used, but exporting the quantity as read from the invoice would clearly be incorrect as 1000 *tonnes* was not what the vendor was billing.

During line pairing, the system will detect this and will make the corresponding adjustment, taking care to pass **1** as the quantity in the scenario above. To activate this check, the following parameter must be set in the *LPR section* of the project configuration:

```
LPR_OP_UOMCheck=YES
```

For systems where a purchase order line has an order unit of measure as well as a price order unit of measure, the parameter below should also be set as follows:

```
LPR_OP_PUOMCheck=YES
```

In the majority of cases, the system will be able to perform this quantity conversion either because the units of measure are universal constants (i.e. kilograms to tonnes) or because the relationship is clear from a comparison of the invoice and purchase order quantities and/or pricing data. The following parameter is used to set the percentage tolerance by which the invoice unit price is allowed to be less or greater than the purchase order unit price in order to infer the same order price unit of measure:

```
LPR_VL_PUOMTolerance=10
```

The recommended default, as shown above, is 10%, although this could potentially be set far higher.

If the system is unable to establish the conversion ratio, or if it is unable to confirm whether the quantity actually requires converting, the line will not be paired. This is most likely in instances where the invoice and purchase order unit of measure differences are custom to a particular material and/or there are significant differences in the pricing.

Example 2:

A construction company orders 100 bags of concrete, expecting to pay $69.90 per bag. The company material number for concrete is **1234A**. The purchase order would appear as follows:

Material No	Description	Quantity	Unit of Measure	Unit Price	Total
1234A	Concrete	100	BAG	69.90	6,990.00

The vendor then issues an invoice for a partial shipment of the concrete, but the vendor is invoicing in **LBS**:

Description	Quantity	Unit of Measure	Unit Price	Total
Concrete	30	LBS	4.72	141.60

Because price differences can occur, converting the invoice quantity from **LBS** to **bags** is not possible to achieve with complete certainty unless it is known how many *LBS* are in each *bag* of that particular type of concrete. As such, the system would fail the line pairing for this example under regular circumstances.

To mitigate this, the AP Packaged Project supports the use a unit of measure conversion table, which, if populated correctly, would allow the example above to succeed. This is discussed further in *Section J2: Working with the Unit of Measure Conversion Table* below.

J2. Working with the Unit of Measure Conversion Table

The unit of measure conversion table is an optional table in the Forms Recognition database that holds the conversion ratio between the base unit of measure and a custom unit of measure for a given material.

This table is called **OFRUOMConversions** and can be created by executing the **WFR_AP_Tables_Create_<*database*>.sql** script provided with the AP Packaged Project. The specification for this table is as follows:

Column Name	Type	Description
MATERIAL	String	Client material number.
BASEUOM	String	Material base unit of measure in the ERP system, which is used to account for all stock of that material in a common unit.
NUMERATOR	Integer	Conversion ratio numerator.
DENOMINATOR	Integer	Conversion ratio denominator.
UOM	String	External unit of measure.
The primary key for this table is defined as MATERIAL, BASEUOM, UOM		

In example 2 above, there are 15 LBS contained within each bag of concrete. Therefore, the table should be populated as follows:

Material	Base UOM	Numerator	Denominator	UOM
1234A	BAG	1	15	LBS

During the conversion, the system will perform a lookup on the table using the external unit of measure read from the invoice (in this case, **LBS**) and the material number read from the purchase order (in this case, **1234A**).

If a record is found, the system will then check whether the order unit of measure on the purchase order (in this case, **BAG**) is equal to the base unit of measure for the material. If it is, the invoice quantity will be converted as per the following calculation:

Invoice Quantity * (*Numerator* / *Denominator*) = *Converted Quantity*

For the given example, that calculation would be:

30 * (**1** / **15**) = 2

This means that *30 LBS* is equal to *2 BAGS*, so **2** is the corresponding quantity expressed in the purchase order unit of measure.

Line pairing will now succeed for the invoice line, and the system will pass a quantity of **2**, a unit of measure of **BAG**, and a total line item value of **141.60** to the downstream ERP system.

The database details are configured in the *UOM section* of the project configuration. The following example shows the delivered configuration, but the database connection string, and the table and column names, may be changed as required:

```
UOM_VL_SQLConnectionGroup=01
UOM_VL_DBTableName=
UOM_VL_DBMaterialNo=MATERIAL
UOM_VL_DBBaseUOM=BASEUOM
UOM_VL_DBNumerator=NUMERATOR
UOM_VL_DBDenominator=DENOMINATOR
UOM_VL_DBExternalUOM=UOM
```

If the database connection cannot be established, or incorrect configuration or table entries are present, then the system will not fail document export, but the line item will not be paired. If line pairing logging is activated, then any error and tracing messages will be written into the AP Packaged Project log file.

J3. Unit of Measure Triangulation

A unit of measure triangulation occurs when the unit of measure on the purchase order line item is not the same as the base unit of measure for the material. In this case, three different units of measure need to be considered:

- The invoice line item unit of measure.
- The purchase order line item unit of measure.
- The base unit of measure for the material.

Example 3:

Consider that the construction company in *Example 2* above decides to order more concrete, although this time they wish to order **10 pallets**. The purchase order would be raised as follows:

Material No	Description	Quantity	Unit of Measure	Unit Price	Total
1234A	Concrete	10	PAL	1,398.00	13,980.00

The vendor continues to bill in **LBS**, and submits their invoice as follows:

Description	Quantity	Unit of Measure	Unit Price	Total
Concrete	900	LBS	4.72	4,2480.00

Within the construction company's ERP system, the base unit of measure for the material remains as **BAG**.

Dealing with this situation requires an additional entry in the unit of measure conversion table to represent the conversion ratio between the material base unit of measure (i.e. **BAG**) and the purchase order unit of measure (i.e. **PAL**).

Each single pallet holds 20 bags, so the additional entry in the conversion table should be as shown in the second line of the table below:

Material	Base UOM	Numerator	Denominator	UOM
1234A	BAG	1	15	LBS
1234A	BAG	20	1	PAL

When converting the invoice quantity, the system first converts it to the quantity in the base unit of measure, then converts it again into the same quantity, but expressed in the purchase order unit of measure. The calculations are as follows:

1. Convert the base unit of measure with the formula:

   Base UOM Quanity = *Invoice Quatity* * (*Numerator* / *Denominator*)

2. Convert to the purchase order unit of measure with the formula:

   Converted Quantity = *Base UOM Quantity* * (*Numerator* / *Denominator*)

Line pairing will now succeed for the invoice line item. A quantity of **3**, a unit of measure of **PAL**, and a total of **4248.00** will be passed to the downstream ERP system.

Under no circumstances should an entry be made in the conversion table as follows:

Material	Base UOM	Numerator	Denominator	UOM
1234A	PAL	1	300	LBS

While this is mathematically correct, it is incorrect from a table population point of view, and may lead to incorrect quantity conversions, as the base unit of measure for the material is not **pallets**. The developer should always check with the client so that the base unit of measure for the material can be established correctly. The content of that column must be consistent for all entries in the table relating to a given material.

J4. Unit of Measure Aliases

Unit of measure aliases are defined to translate a unit of measure as it appears on the invoice into a common code that can be understood by the downstream ERP system.

For example, the vendor may print **EACH** as the unit of measure on the invoice, but the ERP system expects **EA**. Equally, a weight of *tonnes* may appear as **tonne**, **tons**, **ton**, **T** or **MT** on the invoice, but the client uses a unit of measure code of **TO**.

Within the *UOM section* of the project configuration, groups can be set up to map the alias as it appears on the invoice to the client's standard ERP system code. Examples of such entries are shown below:

```
UOM_VL_01_ISOCode=TO
UOM_VL_01_Alias=Tonne,tonnes,ton,MT,T
```

```
UOM_VL_02_ISOCode=CS
UOM_VL_02_Alias=Case,Cases,CAS,CA
UOM_VL_03_ISOCode=KG
UOM_VL_03_Alias=kilogram,kilos,kilo,kilograms
```

The list of aliases can be edited as per project requirements; new units of measure may also be added. Maintaining this list has benefit not only during the line pairing operation, but it can also influence the line item extraction of a unit of measure in a positive manner.

Note: The aliases are not case-sensitive.

Appendix K Configuring Purchase Order Number Validations

K1. Introduction

The AP Packaged Project is able to extract and validate purchase order numbers that relate to data held in ERP system database tables.

Different ERP systems have different ways of organizing data in their underlying databases, which necessitates an alternative approach to data validation depending on what type of ERP system is being used.

As standard, the following types of purchase order numbers are supported for database validation:

1. Standard purchase order numbers (as used by Oracle E-Business Suite and other ERPs)
2. JD Edwards purchase order numbers
3. PeopleSoft purchase order numbers
4. ERP systems where the database record is keyed from the purchase order number and company code

Only one purchase order number validation approach can be used per project file.

If line pairing is to be used, both the header level validation and the line level validation must be configured in the *PON section* and the *LPR section* of the project configuration.

K2. Working with Standard Purchase Order Numbers

Standard purchase orders are those that can be identified uniquely in the downstream ERP system using the purchase order number alone. In other words, the ERP system purchase order header table has a single key field: the purchase order number.

This scenario is used in ERP systems such as Oracle E-Business Suite and other ERPs with the exception of JD Edwards and PeopleSoft.

An example purchase order header table structure (with sample data) reflecting this arrangement would be as follows:

PONUMBER	VENDORID	COMPANYCODE	POTYPE	CURRENCY
1928370	100010	GB01	GD	GBP

When a purchase order number is extracted in the PO Number field, the system can be configured to validate the order against a database, and also to bring back other items of data, such as the vendor ID, the company code, the purchase order type and the currency.

To activate validation of the purchase order via a database, the parameter below must be set as follows:

```
PON_OP_ValidateFromDB=YES
```

To configure the database connection, the SQL connection group number that represents the connection string should be entered against the parameter below (group **01** is used in this example):

```
PON_VL_SQLConnectionGroup=01
```

To complete the configuration, the name of the PO header database table (supplied by the customer) must be specified, along with two mandatory column mappings: the PO number itself and the vendor ID. Additional columns may also be added. The following sample configuration reflects the purchase order header table structure above:

```
PON_VL_DBTableName=PO_HEADER
PON_VL_DBPO=PONUMBER
PON_VL_DBVendorID=VENDORID
PON_VL_DBSiteID=
PON_VL_DBCurrency=CURRENCY
PON_VL_DBCompanyCode=COMPANYCODE
PON_VL_DBStatus=
PON_VL_DBDocType=POTYPE
PON_VL_DBBusinessUnit=
```

The system will now validate any extracted purchase order against the content of this table. If the purchase order is not present, or if a connection to the table cannot be established, or if the column mapping is incorrect, the system will mark the purchase order number field as invalid, and the document will be sent to Verifier.

To default the document company code to the one held on the purchase order, the following parameter must be set:

```
PON_OP_SetCompanyCodeFromPO=YES
```

To validate the purchase order vendor against the current document, the following parameters should be set to **NO**. This should be set in an environment where the invoice vendor must always be the same as the purchase order vendor:

```
VND_OP_IgnorePOVendor=NO
VND_OP_UseASSAIfPOVendorInvalid=NO
```

If the invoice vendor may be different to the purchase order vendor, the parameters should be set as follows:

```
VND_OP_IgnorePOVendor=NO
VND_OP_UseASSAIfPOVendorInvalid=YES
```

The above is the default configuration recommended by Oracle. In Verifier, if the PO vendor differs from the invoice vendor for both options above, the **PO VENDOR IS NOT INVOICE VENDOR** invalid reason must be set. In the case of option 2, the system will strive to set this automatically.

To default the invoice currency as that of the purchase order if no other value has been extracted (which is common for domestic invoices), the following parameter must be set to **YES**:

```
CUR_OP_DefaultPOCurrency=YES
```

The PO Type field (i.e. **MATERIAL** or **SERVICE**), which governs whether line items are required, and also how line pairing is performed, can also be set via the purchase order number lookup.

For example, a customer uses purchase order type **GD** for goods and **SV** for services. To configure the system to set the PO Type field value based on whether **SV** is found in the *POTYPE* column of the purchase order header table, the parameter below should be set as follows:

```
PON_VL_ServicePOTypes=SV
```

Equally, if service purchase order numbers always begin exclusively with **42** or **43**, the PO Type field will be set to **SERVICE** if the purchase order number is successfully validated against the database and the parameter below is set as follows:

```
PON_VL_ServicePOPrefixes=42,43
```

The system can also be configured to pad an extracted purchase order number with leading zeros so that it matches data held in the purchase order header table.

This is controlled via the following parameter:

```
PON_VL_LengthForLeadingZeros=10
```

For example, if this value is set to **10**, and **1928370** is extracted from the invoice, the system will format the purchase order number to **0001928370**.

If the parameter is left blank, no leading zeros will be added.

The purchase order validation can also be extended to include data held at the line item level. This is described further in *Section K6: Purchase Order Line Item Validation* below.

K3. Working with JD Edwards Purchase Order Numbers

JD Edwards purchase orders differ from standard purchase orders as the JD Edwards ERP system uses four keys to identify a purchase order uniquely within the purchase order header table. These keys are:

- The purchase order number
- The company code
- The purchase order type
- The purchase order suffix

Hence, the purchase order header table will have a structure similar to that shown in the example below:

PONUMBER	COMPANYCODE	POTYPE	POSUFFIX	VENDORID	CURRENCY
100233	GB01	OP	00	100010	GBP

As multiple records in this table may have a purchase order number of **100233**, the means by which the database table is queried must be adapted to account for the other key fields. The AP Packaged Project caters for the company code and purchase order type columns, but not the purchase order suffix.

In instances where the purchase order number is guaranteed to be unique across all company numbers, purchase order types and suffixes, the standard purchase order number validation described in *Section K2: Working with Standard Purchase Order Numbers* can be used.

To activate validation of the purchase order number against a JD Edwards table schema, the parameter below must be set to **YES**:

```
PON_OP_JDEPO=YES
```

The purchase order type and company code columns must also be mapped, along with the purchase order number and vendor ID columns:

```
PON_VL_DBPO=PONUMBER
PON_VL_DBVendorID=VENDORID
PON_VL_DBCompanyCode=COMPANYCODE
PON_VL_DBDocType=POTYPE
```

The system will raise an error message if the mapping is insufficient or incorrect.

On the invoice itself, the system expects the vendor to state the purchase order in the format **100233 OP**, in other words the purchase order number and the purchase order type.

When configuring purchase order number formats, the following should be entered for the example above:

```
PON_VL_01_Format=10####OP
PON_VL_01_Ignore=.,
```

This means that the system will be looking for a six-digit number beginning with **10**, which has **OP** at the end.

Note: A common OCR problem is that the letter **O** in **OP** will be read as a zero, hence configuring a second PO number format parameter with the value **10####0P** can be beneficial. The system will auto-correct this to **OP**.

The value **OP** should also be added into the comma-separated list of valid JD Edwards purchase order types:

```
PON_VL_JDEPOTypes=OP
```

When a purchase order number is extracted, the system will automatically separate the purchase order number itself from the purchase order type, both on server side, as well as in Verifier if the whole value is entered into the purchase order number field, and if the PO type is either the first two or last two characters of the string. The purchase order type will be placed into the now mandatory PO Extension field.

In instances where the purchase order type is always going to be **OP** (or another constant), this JD Edwards configuration approach is not appropriate. An alternative approach that uses just the purchase order number and the company code to perform the lookup into the purchase order header table as described in *Section K5: Alternative Purchase Order Number Validation*.

As the Company Code field forms part of the key to reading the purchase order from the database table, it cannot be defaulted from the purchase order. Hence, the strategy to derive the company code must either be to default it based on part of the image filename set during the scanning process (set in the *IMP section* of the project configuration), or to use the associative search engine to derive it based on the bill-to details on the invoice. Oracle recommends the former approach.

The system can also be configured to pad an extracted purchase order number with leading zeros so that it matches data held in the purchase order header table.

This is controlled via the following parameter:

```
PON_VL_LengthForLeadingZeros=10
```

For example, if this value is set to **10**, and **1928370** is extracted from the invoice, the system will format the purchase order number to **0001928370**.

If the parameter is left blank, no leading zeros will be added.

All other aspects of the validation operate in the same way as the standard purchase order number validation as described in *SectionK2: Working with Standard Purchase Order Numbers*.

K4. Working with PeopleSoft Purchase Order Numbers

The PeopleSoft ERP system uses a double key to identify a purchase order number uniquely within the purchase order header table. This key consists of the purchase order number and the purchasing business unit.

A sample purchase order header table is shown below:

PO_ID	BUSINESS_UNIT	VENDOR_ID	VNDR_LOC	PO_TYPE	CURRENCY_CD
1002334	ACME	100010	1000	GD	GBP

PeopleSoft uses a vendor ID and site ID (vendor location) combination in order to identify a single vendor at a single address uniquely.

Additionally, the purchase order header table does not contain a company code or equivalent thereof. The general PeopleSoft equivalent of the company code is the payables business unit. Therefore, the strategy to derive this must either be to default it based on part of the image filename set during the document capture process (set in the *IMP section* of the project configuration), or to use the Associative Search Engine to derive it based on the bill-to details on the invoice.

To activate validation of the purchase order number against a PeopleSoft table schema, the following parameter must be set to **YES**:

```
PON_OP_PeopleSoftPO=YES
```

As a minimum, the purchase order number, vendor ID and purchasing business unit fields must be mapped, as per the example below. The site ID is also recommended:

```
PON_VL_DBPO=PO_ID
PON_VL_DBVendorID=VENDOR_ID
PON_VL_DBSiteID=VNDR_LOC
PON_VL_BusinessUnit=BUSINESS_UNIT
```

The system will raise an error if the mapping is insufficient or incorrect.

On the invoice itself, the system expects the vendor to state the purchase order as **1002334 ACME**, in other words the purchase order number and the purchasing business unit.

When configuring purchase order number formats, the following should be entered for the example above:

```
PON_VL_01_Format=10#####ACME
PON_VL_01_Ignore=.,
```

This means that the system will be looking for a seven-digit number beginning with **10**, which has **BWARE** at the end.

The value **ACME** should also be added into the comma-separated list of valid PeopleSoft business units:

```
PON_VL_PeopleSoftBusinessUnits=BWARE
```

When the purchase order number is extracted, the system will automatically separate the purchase order number itself from the purchasing business unit if the value in the *PO Number* field has the business unit either at the beginning or at the end of the string. The purchasing business unit will be placed into the now mandatory *PO Extension* field.

The system can also be configured to pad an extracted purchase order number with leading zeros so that it matches data held in the purchase order header table. This is configured using the following parameter:

```
PON_VL_LengthForLeadingZeros=10
```

For example, if this value is set to **10**, and **1002334** is extracted from the invoice; the system will format the purchase order number to **0001002334**. If the parameter is left blank, no leading zeros will be added.

All other aspects of the validation operate in the same way as the standard purchase order number validation as described in *SectionK2: Working with Standard Purchase Order Numbers*.

K5. Alternative Purchase Order Number Validation

Alternative purchase order numbers refer to those where a unique purchase order is identified in the purchase order header table by a combination of the purchase order number itself and the company code.

An example database table is shown below:

PONUMBER	COMPANYCODE	VENDORID	PO_TYPE	CURRENCY_CD
1928370	GB01	100010	GD	GBP

To activate the system to use alternative purchase order number validation, the following parameter should be set to **YES**:

```
PON_OP_POKeyIncludesCompanyCode=YES
```

When validating a purchase order number, the system will now access the purchase order header table using both the purchase order number and the company code as they appear in the corresponding project fields.

As a minimum, the purchase order number, company code and vendor ID columns must be mapped as shown below:

```
PON_VL_DBPO=PONUMBER
PON_VL_DBVendorID=VENDORID
PON_VL_DBCompanyCode=COMPANYCODE
```

The system will raise an error if any of the above parameters are missing or incorrectly configured.

Because the company code forms part of the key used to identify the purchase order number, it cannot be derived from reading the purchase order number header table. For that reason, Oracle recommends that it is determined either at the point of document capture and included in the

document filename or it is determined using the Associative Search Engine via the invoice bill-to details.

All other aspects of the validation operate in the same way as the standard purchase order number validation as described in *SectionK2: Working with Standard Purchase Order Numbers*.

K6. Purchase Order Line Item Validation

The system uses purchase order line item information for a number of reasons:

1. To assess whether the purchase order type is **MATERIAL** or **SERVICE** if the information to determine that is held at line item level.
2. To assess whether the invoice is a **MIRA**, where there is a 1:1 relationship between net invoice value and total purchase order value.
3. To perform line pairing.

The structure of the purchase order line item database table will vary depending on the ERP system. An example purchase order line item database table is shown below:

PO NUMBER	LINE	LINE TYPE	MATERIAL	DESCRIPTION	ORDER _QTY	ORD _UOM	UNIT _PRICE	PRICE _UNIT	ORDER _AMT
1928370	10	GOODS	1234A	Concrete	2000	KG	500.00	1000	1000.00

Note that this table has two key fields that uniquely identify a single row: the purchase order number and the purchase order line item number. For PeopleSoft and JD Edwards, additional key fields will also be needed. For PeopleSoft this is the purchasing business unit, and for JD Edwards it is the company code and purchase order type.

To activate validation that incorporates purchase order line items, the parameter below should be set as follows:

```
LPR_OP_GetPOLinesFromDB=YES
```

The SQL connection group representing the database connection string and the purchase order line item table name must also be configured, for example:

```
LPR_VL_SQLConnectionGroup=01
LPR_VL_DBTableName=PO_LINES
```

At the column level, for standard purchase orders, the following parameters must be mapped:

```
LPR_VL_DBPO=PONUMBER
LPR_VL_DBLINE=LINE
```

For JD Edwards purchase orders, the following columns must also be mapped:

```
LPR_VL_DBPO=PONUMBER
LPR_VL_DBLINE=LINE
LPR_VL_DBCOMPANYCODE=
LPR_VL_DBERPPOTYPE=
```

For PeopleSoft purchase orders, the following columns must be mapped:

```
LPR_VL_DBPO=PONUMBER
LPR_VL_DBLINE=LINE
LPR_VL_DBBUSINESSUNIT=
```

For alternative purchase orders, the following columns must be mapped:

```
LPR_VL_DBPO=PONUMBER
LPR_VL_DBLINE=LINE
LPR_VL_COMPANYCODE=
```

Once the key fields have been mapped, further columns representing the actual line item detail itself may be added. The entries below show the correct mapping using the column names in the sample table above:

```
LPR_VL_DBMATERIALNO=MATERIAL
LPR_VL_DBDESCRIPTION=DESCRIPTION
LPR_VL_DBPOQUANTITY=ORDER_QTY
LPR_VL_DBUNITPRICE=UNIT_PRICE
LPR_VL_DBPOTOTAL=ORDER_AMT
LPR_VL_DBUOM=ORD_UOM
LPR_VL_DBPRICEUNIT=PRICEUNIT
LPR_VL_DBITEMCATEGORY=LINETYPE
```

The material number column should be mapped to the column in the purchase order line item table that represents the customer's unique material number for that item.

The price unit column should be mapped if available otherwise the system will assume that it is always **1**.

The item category column should be mapped if the value it contains will help the system determine whether the item is a material or a service PO line, which, in turn, can influence the setting of the *PO Type* field.

For example, if a value of **SRV** or **DSRV** in the line type column denotes that the purchase order line item is a service PO line, the following parameter will switch the *PO Type* field to **SERVICE** if any of the PO lines meet this criterion:

```
PON_VL_ServicePOItemCategories=SRV,DSRV
```

At time of data export, the item category column value is passed back out via the *LineType* parameter.

Equally, the purchase order type can be determined using the order unit of measure. For example:

```
PON_VL_ServicePOUOMs=AU,DAYS,HOURS
```

This means that, if any of the purchase order line item units of measure were either **AU**, **DAYS** or **HOURS**, the *PO Type* field value will be set to **SERVICE**.

The following table shows the columns in addition to the above that may also be mapped, and describes the circumstances under which mapping those fields would be beneficial:

Column(s)	Explanation of Usage
DBMATERIALGROUP	This column represents the material group to which the purchase order line item material belongs. If available, it can be mapped simply to pass this data to a downstream system for a paired line item. It is also used in the automatic tax determination procedure if the material group of the item drives the selection of the correct tax code.
DBTAXCODE DBTAXJURCODE	These two columns represent the tax information that was set when the purchase order was originally raised.

<table>
<tr><th>Column(s)</th><th>Explanation of Usage</th></tr>
<tr><td></td><td>For countries that do not use tax jurisdictions (i.e. where tax rates are set by government at the national level such as within the European Union), the tax code tells the ERP system how to handle tax for the line item in terms of the percentage rate of tax to be charged, as well as describing whether item is a service, tax-exempt, or zero-rated because it is, for example, an EU cross-border transaction. The tax jurisdiction code is not used under this circumstance.

For countries that do use tax jurisdictions (i.e. where tax rates are set at a local level such as in the US, Canada, India and Brazil), the tax code tells the ERP system whether the item is fundamentally subject to tax or not. The accompanying tax jurisdiction code, which is the identification number of a specific tax office, represents the actual percentage rate information.

These columns need only be mapped if the automatic tax determination feature is being used.</td></tr>
<tr><td>DBPUOM</td><td>This column represents the order price unit of measure, which is the unit of measure that is associated with the unit price for the purchase order line item.

It may differ from the regular unit of measure that is associated with the order quantity on the purchase order line.

For example, the customer may raise a purchase order for 1000 each (EA) of product A, but the unit price of $10.00 is set per case (CASE).

If there are 10 EA per CASE, then the PO line would appear thus:

<table>
<tr><th>Description</th><th>Quantity</th><th>UOM</th><th>UnitPrice</th><th>PUOM</th><th>Total</th></tr>
<tr><td>Product A</td><td>1000</td><td>EA</td><td>10.00</td><td>CASE</td><td>1000.00</td></tr>
</table>

The PUOM column must always be mapped in implementations where instances such as the above are possible within the customer's ERP system. Not doing so may cause incorrect invoice quantities to be passed downstream for paired line items.</td></tr>
<tr><td>DBTOTALQUANTITYDELIVERED

DBTOTALVALUEDELIVERED

DBTOTALQUANTITYINVOICED

DBTOTALVALUEINVOICED</td><td>These four columns provide the AP Packaged Project with purchase order line item history data, so that the system knows exactly what has been invoiced and goods-receipted to date, both in terms of quantities and overall values.

It can be extremely beneficial to the success rate of the line pairing operation if this information is available in the purchase order line item table, which may involve creating a view based on the standard line item table, and a separate history table.

Example 1:

An invoice is received for Product A with a quantity of 2, a unit price of $10.00 and an overall line total of $20.00. The corresponding purchase order has two line items, both for Product A with the same quantities and pricing:

<table>
<tr><th>Line</th><th>Description</th><th>OrderQty</th><th>UnitPrice</th><th>Total</th></tr>
<tr><td>1</td><td>Product A</td><td>2</td><td>10.00</td><td>20.00</td></tr>
<tr><td>2</td><td>Product A</td><td>2</td><td>10.00</td><td>20.00</td></tr>
</table>

During line pairing, the system will not be able to make a decision between these identical line items, so line pairing will fail, assuming that the EnableIntegrityCheck parameter is set to YES, otherwise the invoice line would be booked to PO line 2.

However, line item 1 may have an open goods receipt against it, whereas there is no goods receipt against line 2. Hence, line 1 would be the preferable line to book against.

By mapping the additional history columns, it now becomes clear to the system which line to book against as, with this extra information, the lines are no longer identical:</td></tr>
</table>

<table>
<tr><th>Column(s)</th><th>Explanation of Usage</th></tr>
<tr><td></td><td>
<table>
<tr><th>Line</th><th>Description</th><th>OrderQty</th><th>UnitPrice</th><th>Total</th><th>TQD</th><th>TVD</th><th>TQI</th><th>TVI</th></tr>
<tr><td>1</td><td>Product A</td><td>2</td><td>10.00</td><td>20.00</td><td>2</td><td>20.00</td><td>0</td><td>0.00</td></tr>
<tr><td>2</td><td>Product A</td><td>2</td><td>10.00</td><td>20.00</td><td>0</td><td>0.00</td><td>0</td><td>0.00</td></tr>
</table>
In this case, the system will book the invoice to line 1.
Example 2:
A second invoice comes in with identical line item detail as the invoice in example 1. As the first invoice has already been booked on the ERP system, the status of the history will have changed as line 1 will not only have been goods receipted, but fully invoiced as well.
Hence, the purchase order line detail will appear as follows:
<table>
<tr><th>Line</th><th>Description</th><th>OrderQty</th><th>UnitPrice</th><th>Total</th><th>TQD</th><th>TVD</th><th>TQI</th></tr>
<tr><td>1</td><td>Product A</td><td>2</td><td>10.00</td><td>20.00</td><td>2</td><td>20.00</td><td>2</td></tr>
<tr><td>2</td><td>Product A</td><td>2</td><td>10.00</td><td>20.00</td><td>0</td><td>0.00</td><td>0</td></tr>
</table>
In this instance, the system will be able to see that the first purchase order line is already fully invoiced, and the invoice line will be paired to available PO line 2.
In example 1 above, the MIRA condition would be met because the net value of the invoice would be equal to the total value of goods receipts not yet invoiced. By mapping these extra columns, not only would line pairing succeed, but also the user would not be required to validate line item extraction needlessly in Verifier if the system is set to skip line item validation for MIRA invoices. This is enabled in the TAB section of the project configuration.
Example 2 would also meet the MIRA condition if there were a $20 goods receipt value against line 2, but no invoices booked against it.
If the LPR_IgnoreCompletedPOLines parameter is set to YES, any fully booked purchase order lines (i.e. where the total quantity invoiced is greater than or equal to the quantity ordered) will not be considered for line pairing under any circumstances. The first PO line in example 2 illustrates this scenario.
</td></tr>
<tr><td>DBPLANT</td><td>
The plant is a code set in the customer's ERP system that represents the physical location where the purchase order line goods are to be delivered, or where a service is to be performed. For example, it could represent a warehouse or an office building.
The plant column can be mapped simply to pass that data to a downstream ERP system for each paired line item, but the system will require the information if either:

The invoice includes a miscellaneous charge, and miscellaneous charges are set to be output as general ledger account entries, where the corresponding coding string in table OFRMISCACC is driven by the plant on the purchase order.
The automatic tax code determination feature is being used and the country of the invoice company code cannot be used to determine where the goods were delivered.

The specific address details related to each plant code (i.e. the country and the state) can be read from the OFRPLANT database table. Settings in the TAX section of the project configuration control the appropriate data source.
<p>Note: The OFRMISCACC and OFRPLANT tables mentioned above are optional database tables that can be used in conjunction with the AP Packaged Project as described. The WFR_AP_Tables_Create_<<i>database</i>>.sql script provided with the AP Packaged Project creates and populates these tables.</p>
</td></tr>
<tr><td>DBCHARGECODE</td><td>These two columns are made available for implementations involving Oracle E-Business Suite, where the charge code and charge code ID information needs to be</td></tr>
</table>

Column(s)	Explanation of Usage
DBCHARGECODEID	brought into the AP Packaged Project, and then passed back out for each line item where line pairing has succeeded.

K7. Configuring Database Validations Using a Stored Procedure

By default, the system will access purchase order header and line item tables using an SQL *SELECT* call. However, it is possible to use a custom stored procedure instead, which the customer may wish to do for data security purposes. The stored procedure should be written to return a record set in the same way that would have been achieved had a regular SQL *SELECT* statement been executed.

The system can be configured to use a custom stored procedure for both purchase order header and line item validations.

To use a stored procedure for the purchase order header validation, the *UseStoredProcedure* parameter below should be set to **YES**, and the custom stored procedure name and SQL connection group must be specified:

```
PON_OP_UseStoredProcedure=YES
PON_VL_StoredProcedureName=SP_CUSTOM_PO_HEADER
PON_VL_SQLConnectionGroup=01
```

For the purchase order line items, the following parameters apply:

```
LPR_VL_SQLConnectionGroup=01
LPR_OP_UseStoredProcedure=YES
LPR_VL_StoredProcedureName=SP_CUSTOM_PO_LINES
```

Data is passed to the stored procedure via stored procedure parameters, which are defined in the *SPC section* of the project configuration. The parameters can be incoming or outgoing, and can be set either to pass the content of a specific project field, or a hard coded value.

The parameter name should be set to the formal interface parameter name in the stored procedure. Each parameter is assigned a type, and, if that type is *varchar*, a length is assigned as well. The following table lists the parameter types available:

Parameter Type	Description
BOOLEAN	True/false Boolean parameter.
INT	Integer
DATE	Date
DOUBLE	Double
VARCHAR	Variable string The length parameter applies for this type. The default length is 50 characters.

If the parameter type is missing or does not correspond to an entry in the table above, a parameter type of *unknown* will be used.

The parameter value, if set to represent a project field, is case-sensitive and must be the technical name of the field. If the value entered is not the technical name of a field, the system will understand it as a hard-coded value. Input parameters into the stored procedure (i.e. the values passed from the project) should have a direction of **I**, and output parameters coming from the

stored procedure have a direction of **O**. If the direction is missing or invalid, the parameter will be considered as an output parameter.

In the following example, parameters **01**, **02** and **03** are set to pass the purchase order number, the company code and a value of **LO** respectively:

```
SPC_VL_01_ParameterName=P0
SPC_VL_01_ParameterType=VARCHAR
SPC_VL_01_ParameterSize=50
SPC_VL_01_ParameterValue=PONumber
SPC_VL_01_ParameterDirection=I

SPC_VL_02_ParameterName=P1
SPC_VL_02_ParameterType=VARCHAR
SPC_VL_02_ParameterSize=4
SPC_VL_02_ParameterValue=CompanyCode
SPC_VL_02_ParameterDirection=I

SPC_VL_03_ParameterName=P2
SPC_VL_03_ParameterType=VARCHAR
SPC_VL_03_ParameterSize=2
SPC_VL_03_ParameterValue=LO
SPC_VL_03_ParameterDirection=I
```

Appendix L Configuring AP Project Reporting

L1. Introduction

While WebCenter Forms Recognition does not provide a native reporting tool, the AP Packaged Project can be configured to generate comprehensive processing metrics and other data that can be used to create reports.

The **WFR_AP_Reporting_Oracle.sql** script provided with the AP Packaged Project can be (optionally) executed against an Oracle database to create a set of tables for this purpose. If the AP Project Reporting feature is enabled, these tables will be populated with data to allow customers to:

1. Obtain solution key performance metrics.
2. Monitor documents as they move through the system.
3. Identify solution bottlenecks.
4. Report on productivity at the project and client levels.
5. Report on user productivity.

Note: A *sample AP Project Reporting application*, created using Oracle Application Express, is available for download from the Oracle Technology Network. This application is freely distributed for demonstration and educational purposes. As such, it is neither maintained nor supported by Oracle as a licensed product.

L2. Reporting Configuration

The settings required to activate project reporting can be found in the *REP section* and the *SQL section* of the project configuration.

Reporting can be enabled or disabled via the *ConnectToReportingDB* parameter, which should be set to **YES** or **NO** accordingly.

If the parameter is set to **YES**, the first step is to define the SQL connection string used to connect to the reporting database. SQL connection strings are set in the *SQL section* of the project configuration. Within this section, each connection string is specified and assigned to a connection group.

WebCenter Forms Recognition supports multiple database connection strings, as connections to a number of different databases may be required for functions such as data validation, line pairing or data export, as well as the reporting. Each database connection string is assigned a group number that uniquely identifies that connection string. The group number is built into the parameter name of the SQL connection, and, in turn, is used as a reference for all project areas that could require a database connection.

The sample settings below show an Oracle connection string for connection group **02**, but different groups can be configured and used as required:

```
SQL_VL_02_ConnectionString=Provider=OraOLEDB.Oracle.1;Password=myPassword
;Persist Security Info=True;User ID=myUserName;Data Source=ORCL
```

In the *REP section*, this connection group is specified against the *SQLConnectionGroup* parameter so the system knows which connection string to use for the reporting database:

```
REP_OP_ConnectToReportingDB=YES
REP_VL_SQLConnectionGroup=02
```

If no SQL connection group is specified, the system will always default to using group **01**. This applies to all SQL connection groups within the system.

WebCenter Forms Recognition begins the reporting trail for each document upon its initial import into the system, but if the implementation requires that the reporting trail be started earlier (at the point of scan, for example), then the following parameter should be set to **NO**:

```
REP_OP_StartNewRecordForImportedDocument=NO
```

The database key for each reporting record defaults to the filename of the image. Therefore, if the image filename is **12345.tif**, then key for this record in the reporting database will be set to **12345**. If Forms Recognition is not initiating the reporting trail, but rather this is occurring in an upstream system, then the upstream system is responsible for ensuring that the unique ID it has allocated is built into the image filename.

If the image filename comprises of more elements that the unique document identifier, then it is possible to designate a component of the filename as the key for the record in the reporting database, as long as it is bound by an underscore. Assume the image file name follows the naming convention:

<*batch name*>_<*unique id*>_<*company code*>_<*scan date*>**.tif**

The URN is the second component of the filename, so the parameter must be set as follows in the *IMP section* of the project configuration:

```
IMP_VL_URN=COMPONENT2
```

In the *REP section* of the project configuration, the following parameter should also be set:

```
REP_VL_ReportingKey=URN
```

As some customer may insist on a specific naming convention for any tables created within their own databases, the names of the standard AP project reporting tables are configurable via the following parameters:

```
REP_VL_ReportingDBDocumentTable=OFRDOCUMENT
REP_VL_ReportingDBFieldTable=OFRFIELDS
REP_VL_ReportingDBHistoryTable=OFRDOCSTATUS
REP_VL_ReportingDBImageTable=OFRDOCIMAGE
```

The standard table names are shown in the example above. These table names are not case-sensitive.

By default, information is not written to the reporting database from any documents processed using the Designer application, although this can be activated for testing and debugging purposes by setting the following parameter to **YES**:

```
REP_OP_ReportingInDesigner=YES
```

Typically, in production environments, this should always be set to **NO**.

L3. Document Splitting and AP Project Reporting

Within the Verifier application, the user has the ability to split a single document image into two or more document images. If the documents are invoices, then one invoice has now become two or more individual invoices. This means that a corresponding number of new records need to be created in the reporting database.

The system will create these new records at the time where the user manually reclassifies the new individual documents, which will initially have a status of *unclassified* in the AP Packaged Project workflow. Before manual reclassification, the user should ensure that all necessary splits have taken place for a given original image file, otherwise orphan reporting records will remain in the database, which would have the effect of skewing processing statistics.

If usage of the document splitting capability is required, and the system is configured to export data using the standard database, CSV or XML methods, or the file is to be archived into the reporting database, it will not be possible to use the *URN* as mapped to a component of the document filename as the document key for those exports. This is because the URN will remain constant for multiple documents, hence database conflicts will occur, and export files may over-write one another.

Appendix M E-Business Suite Database Views

M1. Introduction

The AP Packaged Project is delivered preconfigured to perform lookups and validations against an E-Business Suite database containing four read-only views, which are described later in this section.

M2. Creating the Views in the E-Business Suite Database

The four views that are configured in the AP Packaged Project are not standard views in the E-Business Suite database and must be created before they can be used by the project. An SQL script file is provided with the project to achieve this, which is located at:

<*Installation Folder*>**\Projects\AP 1007G\DB Scripts\WFR_AP_EBS_Views_Create.sql**

This script is provided as an aid to simplify the implementation of the AP Packaged Project in environments where the customer uses Oracle E-Business Suite as their ERP system, and may be modified as necessary to meet customer requirements. It is not part of the WebCenter Forms Recognition product. For implementations that use an ERP system other than E-Business Suite; it is the responsibility of the customer to provide the appropriate data in a format that can be used by the project.

It is assumed that a schema that will be used by WebCenter Forms Recognition (typically called **AXF**) has already been created in the EBS database, and if this is not the case, this should be done prior to executing the script. The script will create the four views in the **APPS** schema of the EBS database and will grant *SELECT* privileges on them to the **AXF** schema to use them through synonyms.

Simply execute the script file using SQL*Plus or SQL Developer connected to the E-Business Suite as the **APPS** user. During the script execution, you will be prompted to enter the password for the **APPS** user, as well as the username and password for the schema to which you want to grant privileges to perform lookups against the views. This is typically a user called **AXF**, but any schema can be used as preferred.

Note: Because the username and password for the schema will potentially be entered into the project configuration file in plain text it is highly recommended that you use a schema that does not provide UID access to the EBS tables.

M3. The XX_OFR_PO_HEADER_V View

This view provides header-level purchase order data that can be used for PO number validation as described in *Section 3.1.8: Purchase Order Number and PO Extension*.

The view defined in the script provided with the AP Packaged Project excludes incomplete purchase orders from the results.

Column Name	Type	Description
PO_HEADER_ID	NUMBER	Contains the internal ID of the purchase order header.
PO_NUMBER	VARCHAR2(20)	Contains the purchase order number. If PO number validation is enabled, this column should be mapped to

Column Name	Type	Description
		the *DBPO* parameter in the *PON section*.
VENDOR_ID	NUMBER	Contains the internal ID of the vendor that the purchase order was issued to. If PO number validation is enabled, this column should be mapped to the *DBVendorID* parameter in the *PON section*.
VENDOR_SITE_ID	NUMBER	Contains the ID of the vendor site that the purchase order was issued to. If PO number validation is enabled, this column should be mapped to the *DBSiteID* parameter in the *PON section*.
STATUS	VARCHAR2(4000)	Contains the approval status of the purchase order. If PO number validation is enabled, this column should be mapped to the *DBStatus* parameter in the *PON section*.
APPROVED_FLAG	VARCHAR2(1)	Contains a flag value indicating whether the purchase order has been approved. In the E-Business Suite database, this column will typically contain a **Y** if the PO is approved, otherwise it will be blank.
CURRENCY_CODE	VARCHAR2(15)	Contains the three-character ISO currency code for the currency that the purchase order was issued in. If PO number validation is enabled, this column should be mapped to the *DBCurrency* parameter in the *PON section*.
PO_TYPE	VARCHAR2(25)	Contains a value identifying what type of purchase order was issued, for example **STANDARD**, **BLANKET**, etc. If PO number validation is enabled, this column should be mapped to the *DBDocType* parameter in the *PON section*.
CREATION_DATE	DATE	Contains the date that the purchase order was created.
ORG_ID	NUMBER	Contains the organization ID or company code for the organization that issued the purchase order. If PO number validation is enabled, this column should be mapped to the *DBCompanyCode* parameter in the *PON section*.
ORG_NAME	VARCHAR2(240)	Contains the name of the organization or company that issued the purchase order.
ORG_ID_NAME	VARCHAR2(281)	Contains a concatenation of the organization ID and name in the format: <*org_id*> <*org_name*>

M4. The XX_OFR_PO_LINES_V View

This view provides line-level purchase order detail, and can be used for line pairing, as described in *Section 3.2.1: Line Pairing*.

Column Name	Type	Description
PO_DISTRIBUTION_ID	NUMBER	Contains the internal distribution ID for the purchase order line.
PO_HEADER_ID	NUMBER	Contains the internal ID of the corresponding purchase order header.
VENDOR_ID	NUMBER	Contains the internal ID of the vendor that the purchase order was issued to.
PO_NUMBER	VARCHAR2(20)	Contains the purchase order number. This column should be mapped to the *DBPO* parameter in the *LPR section* if the *GetPOLinesFromDB* parameter is set to **YES**.
PO_LINE_ID	NUMBER	Contains the internal ID of the purchase order line.

Column Name	Type	Description
ITEM_DESCRIPTION	VARCHAR2(240)	Contains the item description for the purchase order line. This column should be mapped to the *DBDESCRIPTION* parameter in the *LPR section* if the *GetPOLinesFromDB* parameter is set to **YES**.
MATERIAL_NUMBER	VARCHAR2(25)	Contains the material number for the purchase order line. This column should be mapped to the *DBMATERIALNO* parameter in the *LPR section* if the *GetPOLinesFromDB* parameter is set to **YES**.
UNIT_OF_MEASURE	VARCHAR2(25)	Contains the unit of measure for the purchase order line. This column should be mapped to the *DBUOM* parameter in the *LPR section* if the *GetPOLinesFromDB* parameter is set to **YES**.
UNIT_PRICE	NUMBER	Contains the unit price for the purchase order line. This column should be mapped to the *DBUNITPRICE* parameter in the *LPR section* if the *GetPOLinesFromDB* parameter is set to **YES**.
QUANTITY_ORDERED	NUMBER	Contains the quantity ordered for the purchase order line. This column should be mapped to the *DBPOQUANTITY* parameter in the *LPR section* if the *GetPOLinesFromDB* parameter is set to **YES**.
LINE_TOTAL	NUMBER	Contains the total price for the purchase order line. This is the result of the calculation: <*unit_price*> * <*quantity_ordered*> This column should be mapped to the *DBPOTOTAL* parameter in the *LPR section* if the *GetPOLinesFromDB* parameter is set to **YES**.
QUANTITY_RECEIVED	NUMBER	Contains the quantity that has been received for the purchase order line. This column should be mapped to the *DBTOTALQUANTITYDELIVERED* parameter in the *LPR section* if the *GetPOLinesFromDB* parameter is set to **YES**.
QUANTITY_INVOICED	NUMBER	Contains the quantity that has already been invoiced for the purchase order line. This column should be mapped to the *DBTOTALQUANTITYINVOICED* parameter in the *LPR section* if the *GetPOLinesFromDB* parameter is set to **YES**.
ORG_ID	NUMBER	Contains the organization ID or company code for ship-to organization.
BUSINESS_UNIT	VARCHAR2(25)	Contains the business unit segment from the GL code combination. This column should be mapped to the *DBBUSINESSUNIT* parameter in the *LPR section* if the *GetPOLinesFromDB* parameter is set to **YES**.
LINE_NUM	NUMBER	Contains the number of the purchase order line as it appears on the purchase order. This column should be mapped to the *DBLINE* parameter in the *LPR section* if the *GetPOLinesFromDB* parameter is set to **YES**.
PURCHASE_BASIS	VARCHAR2(30)	Contains the purchase basis of the line item, for example, **GOODS** or **SERVICES**, etc.

M5. The XX_OFR_SUPPLIERS_V View

This view provides supplier information, and can be used to create the Vendor ASSA data pool, as described in *Section 6.2: Create the Vendor ASSA Pool*.

Column Name	Type	Description

Column Name	Type	Description
VENDOR_INDEX	VARCHAR2(81)	Contains a unique vendor identifier, in the format: <*vendor_id*>~<*vendor_site_id*> This column should be identified as the ID column when configuring the Vendor ASSA pool, as described in *Appendix D: Configuring the Vendor ID Field*. It should also be mapped to the *ID* parameter in the *SRC section*.
VENDOR_ID	NUMBER	Contains the internal ID for the vendor. This column does not need to be mapped in the *SRC section*.
VENDOR_NAME	VARCHAR2(240)	Contains the name of the vendor. This column should be mapped to the *Name* parameter in the *SRC section*.
VENDOR_NUMBER	VARCHAR2(30)	Contains the external ID for the vendor. This column should be mapped to the *ExternalVendorID* parameter in the *SRC section*.
VENDOR_SITE_ID	NUMBER	Contains the site ID for the vendor site. This column should be mapped to the *SiteID* parameter in the *SRC section*.
ADDRESS_LINE1	VARCHAR2(240)	Contains the first line (street) of the address for the vendor site. This column should be mapped to the *Address1* parameter in the *SRC section*.
ADDRESS_LINE2	VARCHAR2(240)	Contains the second line (street) of the address for the vendor site. This column should be mapped to the *Address2* parameter in the *SRC section*.
CITY	VARCHAR2(60)	Contains the city for the vendor site. This column should be mapped to the *City* parameter in the *SRC section*.
STATE	VARCHAR2(60)	Contains the state, county or region for the vendor site. This column should be mapped to the *State* parameter in the *SRC section*. For states in the US, this should be expressed as the 2-character state identifier.
POSTAL_CODE	VARCHAR2(80)	Contains the postcode or zip code for the vendor site. This column should be mapped to the *Zip* parameter in the *SRC section*.
COUNTRY	VARCHAR2(100)	Contains the country for the vendor site. This column should be mapped to the *Country* parameter in the *SRC section*. This should be expressed as the 2-character ISO country code.
VAT_REGISTRATION	VARCHAR2(80)	Contains the VAT registration number for the vendor site, if applicable. This column should be mapped to the *VATRegNo* parameter in the *SRC section* if VAT number compliance validation is enabled, as described in *Appendix H: Configuring the VAT Number Compliance Check*.
VENDOR_TYPE	VARCHAR2(30)	Contains a flag that identifies the type vendor type for the site. This can be used to identify a utility vendor by configuring the *UtilityFlag* parameter in the *SRC section* with the column name, and the *UtilityAlias* parameter in the *NUM section* with the value in this column that identifies a utility vendor. For example: `NUM_VL_UtilityAlias=UTILITY` `SRC_VL_UtilityFlag=VENDOR_TYPE`

Column Name	Type	Description
INVOICE_TYPE	VARCHAR2(5)	Contains a value indicating whether the vendor site is expected to issue PO-based invoices, or whether they can legitimately issue invoices that do not reference a purchase order. This column can be mapped to the *InvoiceType* parameter in the *SRC section* if required.
CURRENCY_CODE	VARCHAR2(15)	Contains the 2-character ISO currency code to identify the currency that the vendor site will issue invoices in. This column can be mapped to the *Currency* parameter in the *SRC section*.
ORG_ID	NUMBER	Contains the organization ID or company code of the organization that the vendor site relates to.
ORG_NAME	VARCHAR2(240)	Contains the name of the organization that the vendor site relates to.
ORG_ID_NAME	VARCHAR2(281)	Contains a concatenation of the organization ID and name in the format: <*org_id*> <*org_name*>
SITE_NAME	VARCHAR2(15)	Contains the name of the vendor site.

M6. The XX_OFR_INVOICES_V View

This view provides historical header-level invoice information. It can be used for invoice number format validation as described in *Section 3.1.4: Invoice Number*, and also for checking for duplicate invoices as described in *Section 3.2.8: Duplicate Invoice Number Checking*.

Column Name	Type	Description
VENDOR_ID	NUMBER(15)	Contains the internal ID of the vendor who issued the invoice. Where invoice number format validation is enabled, this column should be mapped to the *VendorID* parameter in the *NUM section*. Where duplicate invoice number detection is enabled, this column should be mapped to the *CDIDBSupplierID* parameter in the *WFR section*.
VENDOR_SITE_ID	NUMBER(15)	Contains the site ID of the vendor site that issued the invoice. Where duplicate invoice number detection is enabled, this column should be mapped to the *CDIDBSupplierSite* parameter in the *WFR section*.
INVOICE_ID	NUMBER(15)	The internal ID of the invoice. Where invoice number format validation is enabled, this column should be mapped to the *RecID* parameter in the *NUM section*.
INVOICE_NUMBER	VARCHAR2(50)	The invoice number as provided by the vendor and as stated on the invoice. Where invoice number format validation is enabled, this column should be mapped to the *InvoiceNumber* parameter in the *NUM section*. Where duplicate invoice number detection is enabled, this column should be mapped to the *CDIDBInvoiceNumber* parameter in the *WFR section*.
INVOICE_DATE	DATE	The date that the invoice was issued by the vendor, as stated on the invoice.
INVOICE_TYPE	VARCHAR2(25)	The type of invoice. For standard invoices, this value is typically **STANDARD**, and for credit notes, it is typically **CREDIT** in an E-Business Suite database. Where invoice number format validation is enabled, values found in this column should be mapped to the *InvoiceAlias* and *CreditAlias* parameters

Column Name	Type	Description
		in the *NUM section*.
ORG_ID	NUMBER(15)	The ID of the organization or company code that the invoice was issued to. Where duplicate invoice number detection is enabled, this column should be mapped to the *CDICBCompanyCode* parameter in the *WFR section*.