Oracle® Forms Recognition
A/P Solution Guide
10g Release 3 (10.1.3.5.0)

June 2011

ORACLE

A/P Solution

10g Release 3 (10.1.3.5.0)

Contents

1 INTRODUCTION

1.1 Purpose

This document, intended for use by a trained Oracle Forms Recognition Administrator, is designed as a product user guide for the A/P Solution. It provides an overview of the A/P Solution configuration to enable the solution to be configured for use in a production environment without the developer needing to be a trained developer.

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

1.2 Scope

This document provides the user with an understanding of the A/P Solution configuration file (.ini file) settings. The following documentation is complementary to this document and the A/P Solution:

- Oracle Forms Recognition Installation User Guide
- Oracle Forms Recognition Runtime Server User Guide
- Oracle Forms Recognition Designer User Guide

2 PROJECT OVERVIEW AND SCOPE

The following section gives an overview of the Oracle Forms Recognition A/P Solution, its architecture and scope.

2.1 Prerequisites

The A/P Solution is a pre-developed project for the processing of invoices (PO, Non-PO and credit notes). The pre-requisites for this A/P solution are Oracle Forms Recognition (minimum version 10.1.3.5.0, Build 4146) to be deployed on the host machine.

2.2 A/P Solution Overview

2.2.1 Supported A/P Documents

The Oracle Forms Recognition A/P Solution currently supports the following document types:

1) Vendor invoices
2) Vendor credit memos
3) 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.

2.2.2 Supported Languages

Oracle Forms Recognition is a language-independent solution and, out of the box, is able to process documents using the Western character set, the Cyrillic character set, and Greek.

The A/P Solution 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, Romanian and Italian.

2.2.3 Project Configuration

As part of the solution, the project requirements can be configured using a project .ini file. These settings are dominant to the property settings with the project itself, but recessive to those configured within the Runtime Server for the defined project. The A/P Solution permits the configuration of the following:

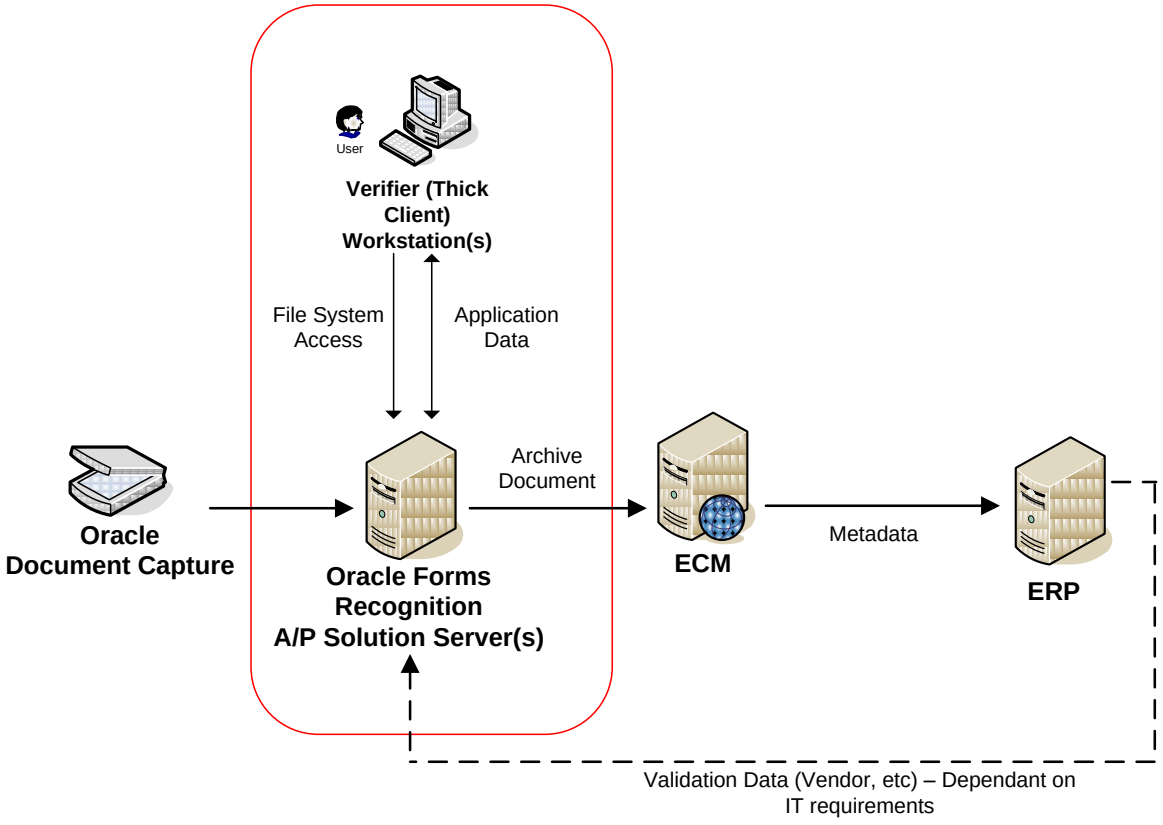
- Business rules relating to pre-defined data fields and document scope
- Database connection settings for validations and reporting
- Connection settings to an ERP database

- Data export settings for non-ERP connections (database connections (Oracle DB or MS SQL) or file export)
- Document archiving and export image file format
- Error messaging, color scheme and presentation of field data to be displayed within Verifier
- Tax code validation and determination

A full list of available configuration settings can be found in section 4.

2.3 A/P Solution Architecture

A typical A/P Solution architecture:



Invoice documents are scanned and passed to Runtime Server. The document is processed by the A/P Solution with the document images and metadata being archived to an ECM system (for example, Oracle I/PM). The Verifier workstation(s) are used for document QA, if required. Once archived to an Oracle ECM system, the documents are processed through a workflow module, which is used for escalating of exceptions and other A/P functions prior to the invoice being booked into the ERP system. Oracle Forms Recognition performs data validation from the ERP

system. The above diagram is for reference only and does not represent any specific IT architecture or architectural limitations of the A/P Solution.

2.4 Local Settings

The A/P Solution project will function on any machine with a language setting which uses the Western alphabet, though English or English (US) is recommended for the server configuration. It also runs independent of whether the machine/system local uses a full stop/period 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), however, this must be changed to a comma or period/full-stop, whichever is appropriate. Dates are handled by the solution internally in a manner that is entirely independent of the local system. The Verifier display and output formats are configurable within the solution.

Amount fields are outputted using a full-stop/period 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 Recommended registry and configuration settings

The following registry settings are recommended when installing Oracle Forms Recognition:

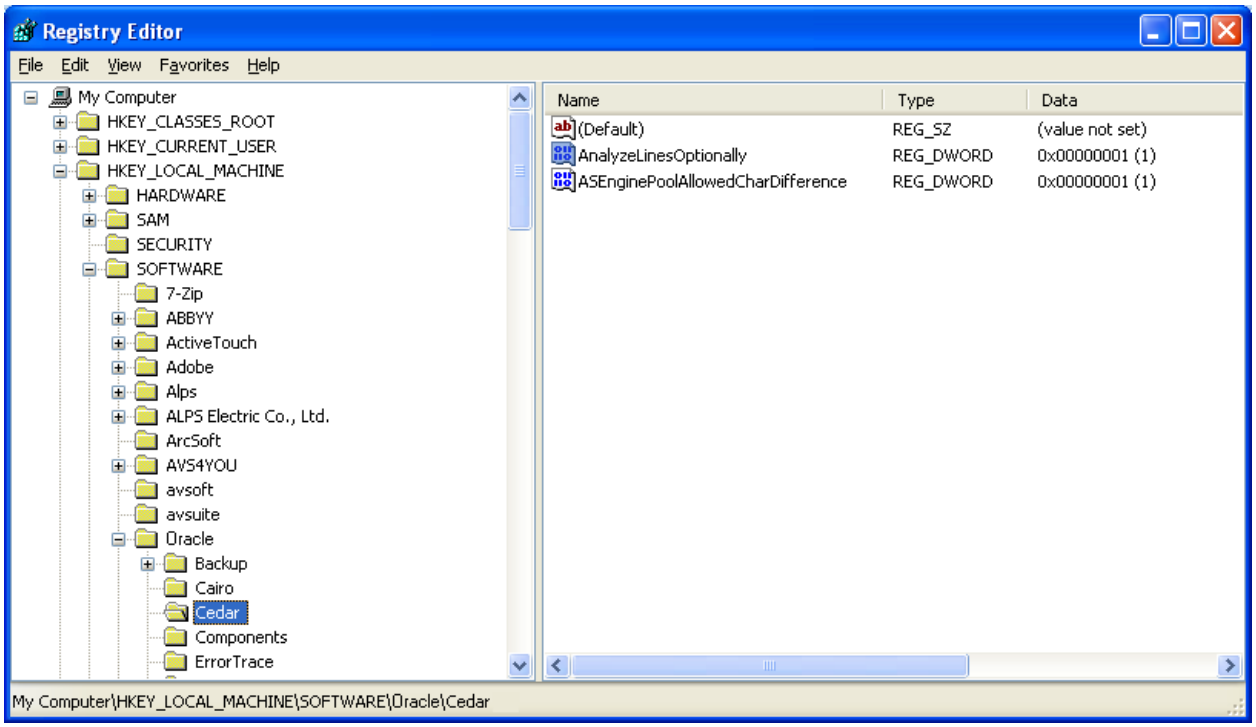
1) AnalyzeLinesOptionally;
2) ASEnginePoolAllowedCharDifference.

‘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 is being used. By default, Oracle Forms Recognition will ‘filter out’ 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.

To set the registry keys, the Windows registry editor should be opened, and a new key ‘Cedar’ created under ‘HKEY_LOCAL_MACHINE\SOFTWARE\Oracle’ if it does not exist already. The two keys should be created as ‘DWORD’ values in the new ‘Cedar’ folder, and both set to a value of ‘1’ with a hexadecimal base.

This is shown in the screenshot below:


Registry Editor
File Edit View Favorites Help
My Computer
HKEY_CLASSES_ROOT
HKEY_CURRENT_USER
HKEY_LOCAL_MACHINE
HARDWARE
SAM
SECURITY
SOFTWARE
7-Zip
ABBYY
ActiveTouch
Adobe
Alps
ALPS Electric Co., Ltd.
ArcSoft
AVS4YOU
avsoft
avsuite
Oracle
Backup
Cairo
Cedar
Components
ErrorTrace
Name
Type
Data
(Default)
REG_SZ
(value not set)
AnalyzeLinesOptionally
REG_DWORD
0x00000001 (1)
ASEnginePoolAllowedCharDifference
REG_DWORD
0x00000001 (1)
My Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Oracle\Cedar

3 PROJECT FIELDS AND FEATURES

The following section provides detail on the fields delivered in the A/P Solution 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 system configuration options in section 4.1.8, 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:

1) Whether line items are required from the invoice or not.
2) How the invoice should be handled downstream.
3) Whether a purchase order number is required.

Within the system configuration settings, a default value for this field can be configured.

For example, 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:

1) Overwrite a NO-PO default and set to PO based upon whether a purchase order number is detected, or a valid purchase order number is detected
2) Overwrite the default based upon an attribute in the ERP vendor master data that would indicate whether NO-PO invoices from this vendor are permitted (for example, the vendor account group, the vendor industry sector, whether the vendor has a purchasing view, etc.)
3) Overwrite the default based on a component of the image filename set by the scanning software where PO and NO-PO invoices are sorted upfront and scanned using different scan jobs

For sample configurations for the invoice type, please refer to Appendix C.

3.1.3 PO Type

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

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

1) The purchase order document type
2) The line type / item category of the purchase order lines
3) The unit of measure on the purchase order lines
4) The prefix of the extracted purchase order number (for example, if service purchase orders all begin with '52' and '523456' is extracted as the purchase order number, then '52' should be entered as the service PO prefix)

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 during document export. The line pairing routine assumes that the purchase order will **NOT** 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 system configuration (see section 4.1.5) and cannot be changed by a user in the Verifier application.

3.1.4 Invoice Number

The invoice number field is mandatory for all documents, but the system can be configured to accept a blank or invalid invoice number in the NUM section (see section 4.1.10) 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.

The validation for point 3) is optional and can be specified in the NUM section of the system configuration options (see section 4.1.10). It can also be specified how many previous invoice numbers the current invoice number should be compared against, and how many hits qualify as

a successful validation. This check is not carried out in the Verifier application as user input is assumed to be correct.

At point of document export, it is possible to configure the system to update the invoice number history table automatically with the results for the current document. As vendors often use a different numbering sequence for invoices and credit notes, the system will account for the document type when performing this check.

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

1) Remove all special characters from an extracted invoice number
2) Remove special characters if they appear at the start or end of the invoice number (Oracle recommends that this setting is always switched on)
3) Retain only a specified set of special characters
4) Remove spaces from within an invoice number
5) Remove any leading zeroes from the invoice number

This feature is available to promote a common standard of invoice number entry to increase the efficacy of a duplicate invoice detection routine in the downstream process.

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 (see section 4.1.11).

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 the 1st February 2009 in Europe.

If the vendor country is mapped in the SRC section (4.1.15), and this country exists in the list of countries where the national date preference is MM/DD/YYYY (for example, 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 is 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 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 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 does not currently support the conversion of dates specified in the Buddhist calendar to the Roman calendar as may be encountered for some invoices from countries such as Thailand.

3.1.6 Company Code

The company code field represents 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 (see section 4.1.14).

The field can be determined either:

1) Via a mapping from a component in the document filename set in the IMP section (see section 4.1.24), 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 system configuration (see section 4.1.4);

3) Via a look-up 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 overwrite any company code determined via steps 1 & 2.

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 system configuration.

3.1.7 Vendor ID / Site ID / Internal Vendor ID

Oracle Forms Recognition employs its unique associative search engine in order to ascertain the invoice vendor.

By pointing Oracle Forms Recognition 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, 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 and NO-PO invoices.

For PO invoices, the vendor is defaulted initially from the purchase order, though this setting can be disabled in the VND section of the system configuration (see section 4.1.16), so that the vendor, that Oracle Forms Recognition 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 (for example, 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 (see section 3.1.28).

For NO-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 (see section 3.1.28).

Oracle Forms Recognition 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.

See Appendix D for more information and examples showing how the various options may be configured.

3.1.8 Purchase Order Number

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 if it matches a valid format as specified in the PON section of the system configuration (see section 4.1.5). Additional information for configuring purchase order number formats can be found in section 6.5.2.

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 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 which 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 system configuration (see section 4.1.16).

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 system configuration (see section 4.1.26), 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.

For more information on how to configure the purchase order number field, please refer to section 10.

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 system configuration (see section 4.1.6), the system administrator can specify words and phrases which 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.

Additional information for configuring bill-to name formats can be found in section 6.5.3.

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

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 system configuration (see section 4.1.20).

The subtotal field may be set as mandatory by setting the 'SubtotalRequired' parameter to 'YES' in the AMT section of the system 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 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

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 system configuration (see section 4.1.20).

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 system configuration (see section 4.1.18). 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.

For more information on configuring the handling of freight, please refer to Appendix E.

3.1.12 Invoice Miscellaneous Charge

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

This field is 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

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 system configuration (see section 4.1.20).

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 system configuration (see section 4.1.18). 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 client 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 system configuration settings.

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

For more information on configuring the handling of miscellaneous charges, please refer to Appendix E.

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 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

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 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 system configuration (see section 4.1.20).

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 the automatic tax determination section of this document 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

The withholding tax amount field captures the portion of the invoice total amount that should be withheld by the client for legal reasons and not paid back to the vendor.

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

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 system configuration (see section 4.1.20).

The withholding tax base amount is set to the full invoice amount (invoice total + withholding tax) minus the total invoice tax amount.

At time of export, withholding tax is outputted as a separate header level field. The invoice total amount is outputted with the withholding tax 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/QST field, although the PST/QST 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

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 system configuration (see section 4.1.20).

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

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 system configuration (see section 4.1.20).

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 system configuration.

3.1.18 Currency

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

In the first instance, Oracle Forms Recognition 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.

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 up the list in the system 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, '$' is found on the document and the vendor is from Canada, hence the system will set the currency to 'CAD'.

Further details on the associated configuration options can be found in section 4.1.13.

3.1.19 Bank Account / Bank Account Code

The bank account number 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 system configuration (see section 4.1.15).

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 system configuration (see section 4.1.27), 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 / POR Subscriber Number

The Payment Order Reference (POR) number is a 27 character transaction ID applied to the invoice by the Swiss Postal Service.

Oracle Forms Recognition 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 configuration (see section 4.1.15). The POR subscriber number in the vendor master takes priority over a POR subscriber number extracted from the invoice.

Neither the POR nor the POR subscriber numbers 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.

As delivered, the extraction of this field is deactivated. The steps required to activate the field are described in Appendix J: Activating the Delivery Note and Payment Reference Fields.

3.1.22 Exchange Rate / Local VAT Amount

If the VAT compliance check is activated in the TAX section of the system configuration (see section 4.1.12), then an exchange rate or VAT amount in local currency must be entered if VAT is being charged in a currency which is not the local currency of the 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 system 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 system configuration (see section 4.1.15) and the value in the vendor master extract column contains the positive value for a utility vendor as specified in the NUM section (see section 4.1.10).

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 over-written either by:

1) The Verifier user via the field drop-down.
2) A component in the document filename being mapped to the priority flag in the IMP section of the system configuration (see section 4.1.24) 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 documents should be sorted according to priority during the scanning process, and then outputted to a different import directory which is swept by an RTS instance that sets the priority of all imported documents to '1'.

3.1.25 Scan Date

This scan date field represents the date upon 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 system configuration (see section 4.1.24). 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 (see section 4.1.25).

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. For more information regarding the mapping of the batch name from the filename, please refer to section 4.1.24.

3.1.27 URN

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

This is tied to the Oracle Forms Recognition 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 Oracle Forms Recognition 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

For more information regarding the mapping of the URN from the filename, please refer to section 4.1.24.

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 reason codes, 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 and NO-PO invoices.	RULE: SETVENDORTOVALID The vendor ID field is set to valid. In the TAB section of the system 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 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.
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 system 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 & 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. No vendor details will be exported. If activated, the VAT registration compliance check will not be carried out.
PO VENDOR <> 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 system 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. e.g. 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.

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

The IVR section of the configuration options (see section 4.1.21) allows 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 of 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 configuration (see section 4.1.21).

3.1.30 Employee ID

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

The Employee ID can be used in a downstream workflow to route the document to a relevant person within the client'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 NO-PO invoice should be posted.

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 system configuration (see section 4.1.4).

3.1.31 Employee Name

The employee name is set by the results of the Associative Search described above.

3.1.32 Line Item Detail

Oracle Forms Recognition will attempt to capture the following information at line item level:

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 system 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.
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. For example, for a line item where 5000 units are invoiced at 100 dollars per 1000 units, the line item total will be 500 dollars. In this case, the price unit is 1000. 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 system configuration (see section 4.1.18 or Appendix F). It cannot be changed by the Verifier user.
VAT Rate	NO	VAT 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 jurisdictions (see Appendix A). 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 look-up, 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 against the primary and secondary rate parameters in the TAX section of the system configuration. The system does not currently have the ability to extract custom vendor tax rate codes at line item level, 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 system configuration options in the TAB section (see section 4.1.17) 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 (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 system configuration (see section 4.1.20).

If the invoice relates to a service purchase order, then the above check is skipped if only the line total column is required.

The information captured within the line items field is used to feed the line pairing feature as described in section 3.2.1.

3.1.33 Vendor VAT Registration Number / 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.

Oracle Forms Recognition is able to carry out this compliance check automatically if VAT compliance checking is activated in the TAX section of the system configuration (see section 4.1.12).

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 system configuration (see section 4.1.15). Additionally, the bill-to company VAT registration number must be available via the company code validation in the CCO section of the system configuration (see section 4.1.14).

The VAT registration number compliance check can be switched on or off 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.

Oracle Forms Recognition 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 system configuration.

3.1.35 Delivery Note

This field is used to capture the vendor delivery note number if stated on the invoice. As delivered, the extraction of this field is deactivated. The steps required to activate the field are described in Appendix J.

3.2 Solution Features

The following section details the features available within the A/P Solution.

3.2.1 Line Pairing

The Oracle Forms Recognition Line Pairing feature leverages the solution'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 Oracle Forms Recognition 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.

Oracle Forms Recognition employs the product's patented fuzzy search technologies to perform the above down to the line item description level. 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 right 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 (see Appendix E for more information).
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, Oracle Forms Recognition can be pointed to a purchase order database or a flat file extract of purchase order line item data.

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 hold:

1) Line pairing is deactivated in the system configuration.
2) Line item extraction is deactivated in the system 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 switched off for utility vendors.
7) The currency of the invoice is not equal to either the currency of the purchase order, or the currency of the company code for which the invoice is intended.
8) The Verifier user has selected an invalid reason of ‘VENDOR NOT FOUND’, and line item extraction is deactivated for that invalid reason
9) The Verifier user has selected an invalid reason of ‘MISSING/INVALID PO’; ‘MISSING/INVALID VENDOR & PO’ or ‘INVOICE AMOUNTS DO NOT ADD UP’.
10) All purchase order line items were marked for deletion.
11) The purchase order has not been released, and line items extraction is deactivated for unreleased purchase orders.

3.2.1.1 Line Pairing For Material Invoices

If the invoice relates to a material purchase order, the system undertakes the following steps before a line item is paired. These steps are as follows:

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 system configuration (see section 4.1.26) with a sufficient distance from the next best possibility, then this is the purchase order line the system selects.

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 converted if required. 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 system configuration (see section 4.1.19).

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 A/P Solution 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’. 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-rate 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 Oracle Forms Recognition 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 (for example, 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 system configuration by settings the 'HandleMiscChargesForServices' parameter to 'YES'.

3.2.1.3 Line Pairing For Third Party Freight Invoices

Within Oracle Forms Recognition, 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 Oracle Forms Recognition 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 system configuration (see section 4.1.18).

The rules for posting third party freight and other miscellaneous charges are described in greater detail in Appendix E.

3.2.2 Automatic Tax Determination & Validation

The A/P Solution 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 completed 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.

For information on configuring the system to operate for countries that do not use tax jurisdictions, please refer to Appendix A.

For information on configuring the system to operate for countries that do use tax jurisdictions, please refer to Appendix B.

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 A/P Solution provides the following standard export options:

1) Export to database tables (see section 7.4)
2) Standard extraction results file (see section 6.6)
3) CSV file output (see section 7.6)
4) XML file output (see section 7.5)
5) Tiff file output (see section 7.2)
6) Fully text-searchable PDF file output (see section 7.3)
7) Integration to archive systems

A user exit is provided for additional export requirements (see section 5.2.2).

Export options can be switched on and off and parameters set via the EXP section of the system configuration (see section 4.1.25).

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 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 Oracle Forms Recognition 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 has connectivity issues 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.

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.

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, irrespective of whether they had been completed beforehand, if required.

3.2.4 Document Management System (DMS) Integration

The A/P Solution 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 reaching Oracle Forms Recognition. In this scenario, Oracle Forms Recognition 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 (see section 4.1.24).

At time of document export, the archive document ID is passed downstream via the Oracle Forms Recognition URN field (see section 3.1.27).

3.2.4.2 Late Archiving

Late archiving means that the image is to be archived after processing in Oracle Forms Recognition.

The standard CSV file output can be configured to produce an import file compliant with Oracle ECM; standard late archiving options are available for integration to document management systems connected to other ERPs.

3.2.5 ERP System Integration

Integration to downstream ERP systems with Oracle Forms Recognition is possible via the following interfaces:

1) Generated XML document
2) Flat file transfer;
3) Export to database staging tables

Sample export flat files can be found in Appendix F.

In the EXP section of the system configuration (see section 4.1.25), the various export options can be activated.

3.2.6 Solution Reporting

The A/P Solution project can populate reporting tables for system reporting. For more information, please refer to section 6.5.1.

4 CONFIGURATION SETTINGS

The project file can be configured via the project .ini file, which resides in the same directory as the project .sdp file. The .ini contains a multitude of settings which controls the way in which the project file behaves. The following sections describe the configuration settings available and how they may be used.

4.1 .Ini File Settings

The .ini is sub-divided into various sections that control different aspects of the project file behavior.

Each .ini file settings parameter is made up of the following nomenclature:

XXX_YY_DDDDD=ZZZ

or:

XXX_YY_NN_DDDDD=ZZZ

XXX = the .ini file section ID (for example, REP, GRL, ITY, EXP etc.)

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

NN = optional .ini file 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.

The following sections can be configured:

4.1.1 GRL Section

This section contains global settings for the project that are used for the purposes of solution reporting.

The following parameters can be set:

Parameter	Type	Description
ProjectName	Freetext	Name of the project

Version	Number	Version number of the project
ClientName	Freetext	Name of the client
VerifierFormStyle	Freetext	Color scheme applied to the Verifier form – options are: 'OFR' – Oracle Form Recognition's color scheme; If any other setting is applied (including blank), then the system will display the default Verifier color scheme (grey form with valid fields marked in green and invalid fields marked in red).

The first three parameters are set primarily from the point of view of reporting, and are stored against all database records created from documents processed through the project.

4.1.2 REP Section

This section contains the configuration settings relating to Oracle Forms Recognition solution reporting.

The following parameters can be set:

Parameter	Type	Description
ConnectToReportingDB	YES/NO	Flag to set whether the project will write out reporting data or not
SQLConnectionGroup	NN	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 module should have the results written to the reporting database
StartNewRecordFor ImportedDocument	YES/NO	If this is set to 'YES', Oracle Forms Recognition will create a new reporting record for each document imported into Designer, removing any old ones for the same document key. If this is set to 'NO', 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 Table	Freetext	Name of the document header table in the reporting database
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
ReportingKey	Freetext	Contains the component to be used as the database table key for the document record. If left blank, the key will be set to the image filename (minus the file extension). If just a component of the filename is required, then this value should be populated with 'URN', then the URN component of the filename should be mapped correctly in the IMP section.

4.1.3 SQL Section

This section contains the SQL connection strings that are used by Oracle Forms Recognition.

The solution supports Oracle and Microsoft SQL Server databases.

Parameter	Type	Description
NN_ConnectionString	Freetext	Connection string for SQL group NN

4.1.4 ASA Section

This section contains settings that control the ASSA pools used for the vendor, employee and company code look-ups in Oracle Forms Recognition. If an ASSA field is not required, it can be removed from the .ini file as long as the ASSA setting is also disabled in the project file (see Appendix G).

The following settings are configurable:

Parameter	Type	Description
Class	Freetext	Name of the class on which the field was created
Fieldname	Freetext	Technical name of the field
AlphaNum	YES/NO	Indicates whether the key field for the pool record is alpha-numeric if set to 'YES'; 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
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 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	System DSN for the ODBC pool import
ImportODBCSelect	Freetext	Select statement used to create the pool
ImportODBCUser	Freetext	User ID used to connect to the database. This can be left blank and specified in the project file if security requires it.
ImportODBCPWD	Freetext	User password to access the database. This can be left blank and specified in the

		project file if security requires it.
AutoImportOption	'FILE', "NONE' or 'ODBC'	Indicates the source from which the pool should be created via the Runtime server (RTS). If set to 'NONE', the pool will not be updated automatically by 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 %

4.1.5 PON Section

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

The following settings are available:

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 (for example, 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 5 digits. It is recommended to define these strings as tightly as possible to optimize the system being able to home in onto the correct value.
NN_Ignore	Freetext	List of characters that may appear in a purchase order number in any position (for example, 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' and '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.
SetCompanyCode FromPO	YES/NO	If set to 'YES', the system will over-write 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 should be validated against a database table. If this validation is activated, the purchase order extracted must exist in the database table and the vendor identified on the invoice must either 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.
POKeyIncludes CompanyCode	YES/NO	This value should be set to 'YES' if the unique key to identify a single purchase order in the PO header database look-up table consists of the purchase order number and the company code in tandem. This should not be set to 'YES' 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.
StoredProcedure Parameters	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 system configuration.
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. This setting is mandatory for database validation.
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 (for example, 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 (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.
LengthForLeading Zeros	Freetext	This is the length that the purchase order number field should be including leading zeros. 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 a 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.
PeopleSoftBusiness Units	Freetext	Comma-separated list of valid PeopleSoft purchasing business units. If 'PeopleSoftPO' is set to 'YES', purchase order numbers will not be extracted 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
ServicePOItem	Freetext	Comma separated list of item categories/line types at the purchase order line item

Categories		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

4.1.6 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.

The following settings are available:

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 (for example, 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 bill-to name. For example, if 'Oracle' is specified as a possible bill-to, 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 was actually 'Oracle 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 (for example, 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.

4.1.7 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 is changed from the delivered default which is:

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

Though more ignore characters may wish to be added covering a wider variety of currency symbols.

The available settings are as follows:

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 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.

4.1.8 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 (for example, CREDIT NOTE, CREDIT MEMO, AVOIR, GUTSCHRIFT).

The following settings are available:

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 (for example, 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 (for example, 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 match at all. The recommended value for this setting is 0.17.

4.1.9 ITY Section

This section includes configuration settings that determine the invoice type field in Oracle Forms Recognition, and whether it is set to 'PO' or 'NO-PO'.

The following settings are available:

Parameter	Type	Description
Default	PO or NPO	Specifies the default value for the field: '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.
SetByVendorCC Exceptions	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 invoice.
NPOValue	Freetext	Denotes the value held in either the document filename or in the vendor extract that represents a NO-PO invoice.
SetToPOIfPOFound	YES/NO	On 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 RTS, if this setting is set to 'YES' and an extracted purchase order is found to exist in a validation database, the invoice type will be set to PO irrespective of any default or vendor specific settings.

4.1.10 NUM Section

This section contains the formatting and validation options available for the invoice number field.

The following settings are available:

Parameter	Type	Description
RemoveAllSpecials	YES/NO	If set to 'YES', Oracle Forms Recognition will remove all special characters from the extracted invoice number.
KeepCertainSpecials	Freetext	Non comma-separated list of special characters that should be retained if 'RemoveAllSpecials' is set to 'YES'.
RemoveSpecialsAtStartAndEnd	YES/NO	If set to 'YES', Oracle Forms Recognition will remove any special characters at the beginning and at the end of an extracted invoice number.
RemoveSpaces	YES/NO	If set to 'YES', Oracle Forms Recognition will remove any spaces from an extracted invoice number.
RemoveLeadingZeros	YES/NO	If set to 'YES', Oracle Forms Recognition will remove any leading zeros 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 will be set to valid and an account number will be required 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 which holds the vendor number. This setting is mandatory.
RecID	Freetext	Column in the database table which 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 which holds the invoice number. This setting is mandatory.
DocumentType	Freetext	Column in the database table which represents the document type: invoice or credit. This setting is mandatory.
InvoiceAlias	Freetext	Value in the document type column in the invoice number history table which denotes an invoice. This setting is mandatory.
CreditAlias	Freetext	Value in the document column in the invoice number history table which 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 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.
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.

4.1.11 DAT Section

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

The following options are available:

Parameter	Type	Description
VerifierOutputFormat	DDMMYYYY or MMDDYYYY	If set to 'DDMMYYYY', Oracle Forms Recognition will display the date in Verifier as DD/MM/YYYY; if set to 'MMDDYYYY', Oracle Forms Recognition will display the date in Verifier as MM/DD/YYYY.
FutureInvoiceDateAllowed	YES/NO	If set to 'NO' and Oracle Forms Recognition has extracted an invoice date 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 of days	If populated, the system will set the invoice date to invalid if it is more than x days in the past from the current date where x is the value of this parameter. 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 (for example, the US)

4.1.12 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 project. This list is optional, but does assist Oracle Forms Recognition 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 project. This list is optional, but does assist in finding the correct tax value on the document.
ActivateVATComplianceCheck	YES/NO	If set to 'YES', the system will activated the VAT registration number compliance check, which 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 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' 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 'VendorVATCheckOnly' 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 'CheckVATCrossBorder' 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 'VendorVATCheckOnly' 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.
TaxFlagException CompanyCodes	Freetext	Comma separated list of company codes that are an exception to the calculate tax flag rule. For example, 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.
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 set in the 'SQL' section. If no connection group is specified, the system will use group '01'.
DBTableName	Freetext	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, as described in section 3.1.34. It should be set to 'NO' if the client is not using the solution to process Brazilian nota fiscal documents.
DeriveShipToFrom CompanyCode	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 used 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.
NN_IfTaxState	Freetext	Tax code for use if the vendor is from a state listed in 'NN_VendorState', or the goods were shipped to a state listed in 'NN_ShipToState' 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, else the 'IfTax' code will be used.
NN_IfNoTaxState	Freetext	Tax code for use if the vendor is from a state listed in 'NN_VendorState', or the goods were shipped to a state listed in 'NN_ShipToState' 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, else 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 NN_TaxCode 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 NN_Country 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.

4.1.13 CUR Section

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

The following options are available:

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 (for example, 01, 02, 03 etc…)
NN_Alias	Freetext	Comma separated list of aliases for the currency represented by NN_ISOCode For example, pounds, sterling for a 'GBP', 'US dollars' or 'US funds' for 'USD', 'Swiss Francs' for 'CHF'. If any items on this list are found on the document, then the extracted invoice currency is set to the currency in NN_ISOCode
NN_AmountPrefix	Freetext	Comma separated list of values that may precede or follow an amount on the invoice that represent the currency specified in NN_ISOCode For example, US$ would be the prefix for US$1000.00
NN_Symbol	Freetext	Symbol associated with the currency held in NN_ISOCode (for example, $). This should only be set to a special character, not a single letter.
NN_Country	Freetext	Country associated with the currency held in NN_ISOCode. If the symbol for the currency settings group is found on the document and 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 system 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.
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 Oracle Forms Recognition.
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 look-up 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 look-up is to be performed.
Activate ExtendedValidation	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 below in the order specified: 1) Permitted global currencies (see 'GlobalCurrencies' parameter below); 2) The vendor currency if present in the vendor extract; 3) The currency associated with the vendor country of origin; 4) The currency of the company code; 5) 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 look-ups 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 in so far as a valid currency is entered. The extended validation is only carried out for invoice classified to the generic node.
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'.

4.1.14 CCO Section

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

The options available are as follows:

Parameter	Type	Description
AllowBlank	YES/NO	Controls whether the company code is mandatory if the value is set to 'NO'.
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 used group '01'.
DBTableName	Freetext	Name of the company code database table. This setting is mandatory if a database look-up 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 look-up 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.
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.

4.1.15 SRC Section

This section holds the mapping between columns in the vendor extract and the values used internally within the Oracle Forms Recognition project.

The following settings are available:

Parameter	Type	Description
ID	Freetext	ASSA column name denoting the vendor ID. 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: Vendor ID * 1000000 + 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 system configuration. The system will raise a configuration error if no delimiter is specified, or it is more than one character, or it 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 SRC_VL_SITEID, and the vendor ID stem must be mapped to SRC_VL_EXTERNALVENDORID. If the ERP system uses an external vendor ID (for example, the supplier number in Oracle Financials), then this value should be mapped to SRC_VL_EXTERNALVENDORID 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. More detail regarding the configuration and set-up of the Vendor ID field is described in Appendix D.
SiteID	Freetext	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	ASSA column name denoting the vendor name.
Address1	Freetext	ASSA column name denoting the first line of the vendor's address.
Address2	Freetext	ASSA column name denoting the second line of the vendor's address.
City	Freetext	ASSA column name denoting the vendor's city of origin.
Zip	Freetext	ASSA column name denoting the vendor zip/postal code.
State	Freetext	ASSA column name denoting the vendor's state/region.
Country	Freetext	ASSA column name denoting the vendor's country of origin.
POBox	Freetext	ASSA column name denoting the vendor's PO Box.
POBoxZip	Freetext	ASSA column name denoting the postal/zip code that relates to the vendor's PO Box.
EUMember	Freetext	ASSA column name denoting whether the vendor's country of origin is a member of the European Union.
Currency	Freetext	ASSA column name denoting the currency for the vendor's country of origin.
TaxID1	Freetext	ASSA column name denoting the vendor primary tax ID.
TaxID2	Freetext	ASSA column name denoting the vendor secondary tax ID.
VatRegNo	Freetext	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	ASSA column name denoting the vendor's tax jurisdiction code (US).
TelNo	Freetext	ASSA column name denoting the vendor's telephone number.
InvoiceType	Freetext	ASSA column name denoting whether the vendor is permitted to submit a NO-PO invoice.
PaymentMethods	Freetext	ASSA column name denoting the vendor's payment methods.
BankDetails	Freetext	ASSA column name denoting the vendor's bank account details This should be a colon-separated list in the format: BankAccount,SortCode,ERPBankAccountCode A sortcode is the US equivalent of a routing number.
WithholdingTax Details	Freetext	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	ASSA column name denoting a comma-separated list of company codes for which the vendor is valid.
UtilityFlag	Freetext	ASSA column name denoting whether the vendor is a utility vendor or not.
PORSubscriberNo	Freetext	ASSA column name denoting the vendor's Post Office Reference number as used in Switzerland.
ExternalVendorID	Freetext	ASSA column name denoting the vendor's external ID (for example, 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.
EUMemberAlias	Freetext	Contains the value that denotes a positive identification of the vendor as an EU state member.

4.1.16 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 / Vendor Extract. Oracle recommends that this setting always be set to 'YES'.
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 <> 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 <> 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 'Ignore POVendor' is set to 'NO'.

DefaultCountry	Freetext	If no country column is available in the vendor extract used by the VendorASSA 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-code (e.g. United States = 'US', United Kingdom = 'GB', Germany = 'DE' etc…)
UseBillToBasedVendor Extraction	YES/NO	If this value is set to 'YES' and a bill-to name field has been captured from the 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.
FilterVendorsByCompanyCode	YES/NO	Set this parameter to 'YES' to support filtering of vendors by company code. This filtering means that after extraction and when performing a vendor search in the Verifier form, all vendor candidates that do not belong to the invoice's company code are discarded. Note that this parameter requires that the company code be assigned to the invoice at scan or import and supplied to each vendor in the vendor pool.

4.1.17 TAB Section

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

The available options are:

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.
LineTotalOnlyFor Service	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 which 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 documents will multiple backing sheets that the system does not need to evaluate as potential line items that require extraction.

4.1.18 MSC Section

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

The following settings are available:

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.
HandleMiscChargesForServices	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 used group '01'.
DBTableName	Freetext	Name of the database table containing the GL coding strings to be used for miscellaneous charges. This table should follow the layout of the MiscAcc table definition,
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 Oracle Forms Recognition header field which 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 (for example, FRT for freight in Oracle).
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_ValidConditions	Freetext	This is a comma-separated list of valid condition types that relate to the charge group NN that could appear as planned charges against a purchase order line item.
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 group were found on the invoice, the system will handle these charges as unplanned if this setting is set to 'YES'
NN_BookToGL AccountIfNoPlanned	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_GetCostObject FromPOLine	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_ DefaultProfit Center	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.

4.1.19 UOM Section

This section contains mapping information between a unit of measure and how it 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.
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 used group '01'.
DBTableName	Freetext	Name of the unit of measure conversion database table. This setting is mandatory if a database look-up 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 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

4.1.20 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 on the system 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.

The following tolerances are available:

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 (there are no active 'cent' or 'penny' sub-units), for example, the Japanese Yen or the Hungarian Forint. For example, 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, then the formatting will stay at '10.40'.

4.1.21 IVR Section

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

The available settings are:

Parameter	Type	Description
DefaultText	Freetext	Default invalid reason (for example, 'NONE')
DefaultExportCode	Freetext	Export code associated with the default invalid reason (for example, 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. The available rules are: SETVENDORTOVALID - sets the vendor field to valid SETPOTOVALID – sets the purchase order number field to valid; no line pairing is carried out SETVENDORANDPOTOVALID – sets the purchase order and vendor number fields to valid ALLOWNONPOVENDOR – allows a vendor ID to pass even if it is unconnected to the purchase order as long as the vendor ID exists SETAMOUNTSTOVALID – sets the amount fields and the table to valid; no line pairing is carried out THIRDPARTYFREIGHT – sets the vendor number field to be valid as long as it exists and processes the document according to the third party freight rules during line pairing.

		SETVENDORANDPOTOVALID – sets the purchase order and vendor number fields to valid; no line pairing is carried out. NONVATCOMPLIANT – sets the local VAT amount, exchange rate and vendor & bill-to VAT registration number fields to valid. STOCKINVOICE – sets the purchase order number field to valid, line pairing is still carried out based on purchase orders set in 'UserExitLinePairingPOs'.
NN_VerifierDisplay	Freetext	Invalid reason message displayed in Verifier for group NN.
NN_ExportCode	Freetext	Invalid reason code exported by Oracle Forms Recognition if the invalid reason field is set belonging to group NN.

4.1.22 ERR Section

This section contains the field error messages displayed to the user through the Verifier application. Each setting takes the format:

```
ERR_VL_NNN=[Error Text]
```

…where NNN is the error code number.

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

4.1.23 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.

In common with the error message section, only the text associated with each message should be changed.

4.1.24 IMP Section

This section contains settings revolving around document import, specifically the mapping of values contained within the image filename to fields in Oracle Forms Recognition. This provides a simple means to pass data to Oracle Forms Recognition from an upstream system.

Filename components should be separated by an underscore (for example, COMPONENT1.tif, COMPONENT1_COMPONENT2.tif, COMPONENT1_COMPONENT2_COMPONENT3.tif)

For example, `IMP_VL_ScanDate=COMPONENT1` for the file 12022008_1234_123456.tif would put 12022008 into the ScanDate field.

The following settings are available

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, 'ODC', 'ODDC')
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 or YYYYMNDD.

4.1.25 EXP Section

This section holds the configuration settings relating to the Oracle Forms Recognition data export options.

The following settings can be configured:

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.
OutputStandard ResultsFile	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. 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 which is used as the default should no export directory be set in RTS.
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 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.
TiffDPI	Number	Specifies the DPI of the outputted tiff image (for example, 300) In case of a blank or invalid entry, the default tiff resolution is 300 dpi.
TiffFormat	Freetext	Compression format of the outputted tiff file. Options 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.
RedactInvoice Number	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. If left blank, or set to anything else, the filename will be set to the same name as the document filename.
OutputDateFormat	DDMMYYYY MMDDYYYY YYYYMMDD	Output date format for export. 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/slash, a period/full-stop or a forward slash) This 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 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 column in the SRC 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	INSERT or UPDATE	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 existing record with the same database key. There is no default setting.
DBHCDocumentType	Freetext	Name of the header export database column which holds the document type.
DBHCInvoiceType	Freetext	Name of the header export database column which holds the invoice type.
DBHCPOType	Freetext	Name of the header export database column which holds the PO type.
DBHCERPDocType	Freetext	Name of the header export database column which holds the ERP system document type.
DBHCCompanyCode	Freetext	Name of the header export database column which holds the company code.
DBHCInvoiceNumber	Freetext	Name of the header export database column which holds the invoice number.
DBHCInvoiceDate	Freetext	Name of the header export database column which 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 which holds the vendor ID.
DBHCInternalVendorID	Freetext	Name of the header export database column which holds the internal vendor ID.
DBHCSiteID	Freetext	Name of the header export database column which holds the vendor site ID.
DBHCVendorAccount Group	Freetext	Name of the header export database column which holds the ERP vendor account group.
DBHCVendorName	Freetext	Name of the header export database column which holds the vendor name.
DBHCBillToName	Freetext	Name of the header export database column which holds the bill-to name.

DBHCPO	Freetext	Name of the header export database column which holds the purchase order number.
DBHCTax	Freetext	Name of the header export database column which holds the invoice tax amount.
DBHCWithholdingTax	Freetext	Name of the header export database column which holds the invoice withholding tax amount.
DBHCMisc	Freetext	Name of the header export database column which holds the invoice miscellaneous charge amount at header level.
DBHCDiscount	Freetext	Name of the header export database column which holds the invoice header level discount.
DBHCInvoiceTotal	Freetext	Name of the header export database column which holds the invoice total amount.
DBHCPST	Freetext	Name of the header export database column which holds the PST/QST tax amount.
DBHCICMS	Freetext	Name of the header export database column which holds the Brazilian ICMS tax amount.
DBHCUnplannedFreight	Freetext	Name of the header export database column which holds the unplanned miscellaneous charge amount from the invoice. This will default to the freight amount specified on the invoice is line pairing is not carried out, or no configuration exists for miscellaneous charges.
DBHCCurrency	Freetext	Name of the header export database column which holds the invoice currency.
DBHCPORNumber	Freetext	Name of the header export database column which holds the Post Office Reference (POR) number. This is used in Switzerland.
DBHCPORSubscriberNo	Freetext	Name of the header export database column which holds the Swiss POR subscriber number for the vendor.
DBHCPaymentReference	Freetext	Name of the header export database column which holds the invoice payment reference (for example, the Norwegian KID number).
DBHCBankAccount	Freetext	Name of the header export database column which holds the vendor's designated bank account.
DBHCBankAccountCode	Freetext	Name of the header export database column which holds the ERP system code for the vendor's designated bank account.
DBHCExchangeRate	Freetext	Name of the header export database column which holds the invoice currency exchange rate.
DBHCEmployeeID	Freetext	Name of the header export database column which holds the ERP system employee ID.
DBHCEmployeeName	Freetext	Name of the header export database column which holds the name of the employee.

DBHCAccountNumber	Freetext	Name of the header export database column which holds the account number for which the invoice relates (for example, as found on utility bills).
DBHCScanDate	Freetext	Name of the header export database column which 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 which holds the name of the batch into which the invoice was scanned.
DBHCPriorityFlag	Freetext	Name of the header export database column which holds the invoice priority flag (YES or NO).
DBHCURN	Freetext	Name of the header export database column which holds the document URN.
DBHCInvalidReason	Freetext	Name of the header export database column which holds the text for the invalid reason as set by the Verifier user.
DBHCInvalidReasonCode	Freetext	Name of the header export database column which holds the export code relating to the invalid reason as set in the IVR section.
DBHCPaymentMethod	Freetext	Name of the header export database column which holds the vendor payment method code.
DBHCWithholdingTaxCode	Freetext	Name of the header export database column which holds the vendor withholding tax code.
DBHCAlternatePayee	Freetext	Name of the header export database column which holds the alternate payee associated with the vendor.
DBHCPermittedPayee	Freetext	Name of the header export database column which holds the permitted payee associated with the vendor.
DBHCStatus	Freetext	Name of the header export database column which holds the current document status. This is subsequently overwritten with the new document status (document exported from Oracle Forms Recognition), the corresponding value of which is set in DBStatusExported.
DBHCERPPOType	Freetext	Name of the header export database column which holds JD Edwards purchase order type code for the purchase order captured at header level
DBHCBusinessUnit	Freetext	Name of the header export database column which holds the PeopleSoft purchasing business unit for the purchase order number captured at header level
DBHCDeliveryNote	Freetext	Name of the header export database column which holds the delivery note number of the vendor
DBStatusExported	Freetext	Value or code that denotes a document successfully exported from Oracle Forms Recognition.
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 (for example, 1, 2, and 3 etc…)
DBTablePO	Freetext	Name of the column in the line item export database table which 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 which 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 which holds the invoice line item description.
DBTableMaterialNo	Freetext	Name of the column in the line item export database table which holds the invoice line item material number.
DBTableMaterialGroup	Freetext	Name of the column in the line item export database table which holds the invoice line item material group.
DBTableQuantity	Freetext	Name of the column in the line item export database table which holds the invoice line item quantity.
DBTableUOM	Freetext	Name of the column in the line item export database table which holds the invoice line item order unit of measure.
DBTableUnitPrice	Freetext	Name of the column in the line item export database table which holds the invoice line item unit price.
DBTablePUOM	Freetext	Name of the column in the line item export database table which 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 which holds the invoice line item total amount.
DBTableTaxCode	Freetext	Name of the column in the line item export database table which holds the invoice line item tax code.
DBTableTaxJurCode	Freetext	Name of the column in the line item export database table which holds the invoice line item tax jurisdiction code.
DBTableConditionType	Freetext	Name of the column in the line item export database table which holds the invoice line item condition type.
DBTableCondition StepNo	Freetext	Name of the column in the line item export database table which holds the invoice line item condition step number.
DBTableCondition Count	Freetext	Name of the column in the line item export database table which holds the invoice line item condition count.

DBTableFreightVendor	Freetext	Name of the column in the line item export database table which holds the invoice line item freight vendor.
DBTableGRDocNo	Freetext	Name of the column in the line item export database table which 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 which 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 which 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 which holds the service entry sheet to which the invoice line relates.
DBTableSheetItem	Freetext	Name of the column in the line item export database table which holds the item number of the service entry sheet to which the invoice line relates.
DBTableSubDeb Indicator	Freetext	Name of the column in the line item export database table which 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 which holds the invoice line item type or category.
DBTableChargeCode	Freetext	Name of the column in the line item export database table which holds the invoice line item charge code.
DBTableChargeCodeID	Freetext	Name of the column in the line item export database table which holds the invoice line item charge code ID.
DBTableDistillerLine	Freetext	Name of the column in the line item export database table which holds the original line number for the item in the table captured by Oracle Forms Recognition. This value is only populated if line pairing is switched on and then only for lines where the line pairing was not successful. Internally, Oracle Forms Recognition table lines use a zero-based index. i.e. the first line item in the Oracle Forms Recognition table is line zero internally. This field is populated with an incremented value so that a value of 1 will be output for line item 1, line 2 for line item 2 etc…
DBTablePlant	Freetext	Name of the column in the line item export database table which holds the plant to which the invoice line item relates.
DBTableCompanyCode	Freetext	Name of the column in the line item export database table which holds the company code
DBTableERPPOType	Freetext	Name of the column in the line item export database table which holds the JD Edwards purchase order type
DBTableBusinessUnit	Freetext	Name of the column in the line item export database table which holds the PeopeSoft purchasing business unit

DBTableTaxRate	Freetext	Name of the column in the line item export database table which 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 (for example, 1, 2, 3, etc.)
DBGLTableGLAccount	Freetext	Name of the column in the general ledger export database table which holds the general ledger account number.
DBGLTableCompanyCode	Freetext	Name of the column in the general ledger export database table which holds the company code for the general ledger entry.
DBGLTableCostCenter	Freetext	Name of the column in the general ledger export database table which holds the cost center for the general ledger entry.
DBGLTableInternalOrder	Freetext	Name of the column in the general ledger export database table which holds the internal order for the general ledger entry.
DBGLTableWBSElem	Freetext	Name of the column in the general ledger export database table which holds the Work Breakdown Structure for the general ledger entry.
DBGLTableNetwork	Freetext	Name of the column in the general ledger export database table which holds the network for the general ledger entry.
DBGLTableActivity	Freetext	Name of the column in the general ledger export database table which holds the network activity for the general ledger entry.
DBGLTableProfitCenter	Freetext	Name of the column in the general ledger export database table which holds the profit center for the general ledger entry.
DBGLTableSalesOrder	Freetext	Name of the column in the general ledger export database table which holds the sales order for the general ledger entry.
DBGLTableSalesOrderItem	Freetext	Name of the column in the general ledger export database table which holds the sales order item for the general ledger entry.
DBGLTableBusinessArea	Freetext	Name of the column in the general ledger export database table which holds the business area for the general ledger entry.
DBGLTableTaxCode	Freetext	Name of the column in the general ledger export database table which holds the tax code for the general ledger entry.
DBGLTableTaxJurCode	Freetext	Name of the column in the general ledger export database table which holds the tax jurisdiction code for the general ledger entry.
DBGLTableTotal	Freetext	Name of the column in the general ledger export database table which 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 which 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 default export path 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	URN [blank]	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. 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 '.' Is not required For example, XML = .XML; TXT = .txt If left blank, the file extension will default to 'XML'.
XMLStatusExported	Freetext	Value or code that denotes a document successfully exported from Oracle Forms Recognition. 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. For example, <MyFileHeader> This value will default to 'Document' if nothing else is set.
XMLDocClass	Freetext	This denotes the tag to be used to mark the Document Class in the document header section of the XML output file. If set to 'DocClass', the output will be: For example, <DocClass>[Document Class]</DocClass> If this parameter is left blank, the 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 system configuration in the document header section of the XML output file. If set to 'ScanDate, then the output will be: For example, <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 (i.e. DD/MM/YYYY, MM-DD-YYYY, YYYYMMDD) is configured using the 'EXP_VL_ OutputDateFormat' and 'EXP_VL_OutputDateSeparator' parameters.
XMLURN	Freetext	This denotes the tag to be used to mark the document URN (as mapped in the IMP section of the system configuration) in the document header section of the XML output file. If set to 'URN, the output will be: For example, <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 system configuration) in the document header section of the XML output file. If set to 'BATCHNAME', the output will be:

		For example, <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 batch name (as mapped in the IMP section of the system configuration) in the document header section of the XML output file. If set to 'BATCHNAME', the output will be: For example, <BATCHNAME>[Document Batch Name]</BATCHNAME> If this parameter is left blank, the document batch name 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 or the file path to the image in a storage directory. This is dependent on settings in the REP section of the system configuration. If set to 'DocLink, the output will be: For example, < 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 EXP_VL_XMLStatusExported) in the document header section of the XML output file. If set to 'STATUS, the output will be: For example, <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. For example, <InvoiceHeader> This value will default to 'InvHeader is nothing else is set.
XMLHCxxx	Freetext	This denotes the tag to be used to mark the invoice header output field represented by 'xxx'. For example: XMLHCInvoiceNumber is the parameter for the invoice number. If this parameter is set to 'InvNo', and '1234' is extracted as the invoice number, then the following line will be written into the invoice header section of the XML file. For example, <InvNo>1234</InvNo> If this parameter is left blank, the field will not be written into the invoice header. The following fields are available for output: XMLHCDocumentType – document type (INVOICE or CREDIT) XMLHCInvoiceType – invoice type (PO or NO-PO) XMLHCPOType – PO type (MATERIAL or SERVICE) XMLHCERPDocType – ERP system document type XMLHCCompanyCode – invoice company code XMLHCInvoiceNumber – invoice number

		XMLHCInvoiceDate – invoice date XMLHCVendorID – vendor ID XMLHCInternalVendorID – internal vendor ID XMLHCSiteID – site ID XMLHCVendorAccountGroup – vendor account group XMLHCVendorName – name of the vendor XMLHCBillToName= bill-to name XMLHCPO- purchase order number XMLHCTax – total tax amount XMLHCWithholdingTax – withholding tax amount XMLHCMisc – miscellaneous charge amount XMLHCDiscount – discount amount XMLHCInvoiceTotal – invoice total XMLHCPST – PST/QST amount (for Canada) XMLHCICMS – Brazilian ICMS tax amount XMLHCUnplannedFreight - unplanned freight amount XMLHCCurrency - invoice currency XMLHCPORNumber – POR reference number XMLHCPORSubscriberNo – POR subscriber number XMLHCPaymentReference – payment reference XMLHCBankAccount – bank account number XMLHCBankAccountCode – ERP system bank account code XMLHCExchangeRate – exchange rate XMLHCEmployeeID – employee ID XMLHCEmployeeName – employee name XMLHCAccountNumber – account number XMLHCInvalidReason – invalid reason text XMLHCInvalidReasonCode – invalid reason code XMLHCPaymentMethod – vendor payment methods XMLHCWithholdingTaxCode - withholding tax codes XMLHCAlternatePayee – alternate payees XMLHCPermittedPayee – permitted payees XMLHCERPPOType – JD Edwards PO type XMLHCBusinessUnit – PeopleSoft purchasing business unit XMLHCDeliveryNote – delivery note number

4.1.26 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.

The available options are:

Parameter	Type	Description
DoLinePairing	YES/NO	Flag to indicate whether the line pairing operation should be carried out at the point of document export.
DoLinePairingFor Service	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.
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 was 'BROWN HATS', then a 100% match would be achieved if the invoice line item description was '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. The default value is 0%. The recommended setting for this value is 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 order unit of measure differs from the PO order 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 look-up to the unit of measure conversion table specified in the UOM section of the system 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 look-up is not configured correctly, or the relevant entry is missing, the line item will not be paired.
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 UOM Check is set to YES. The recommended setting is 10.
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 FindGR	YES/NO	This parameter is used in conjunction with the above parameter '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'.
RequirePODetails ForMultipleMaterials	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.
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: 1) A workflow component sits between Oracle Forms Recognition and the downstream ERP system; 2) Oracle 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's a one-to-one relationship between the invoice and purchase order.
ActivateLog	YES/NO	If set to 'YES', the system will write out a log file containing trace entries for the line pairing operation. This is written into the standard Oracle Forms Recognition log file.
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 (for example, 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 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 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 system configuration.
DBPO	Freetext	Name of the column in the purchase order line database table which holds the purchase order number.

DBLine	Freetext	Name of the column in the purchase order line database table which holds the purchase order line number.
DBMaterialNo	Freetext	Name of the column in the purchase order line database table which holds the purchase order line item material number.
DBMaterialGroup	Freetext	Name of the column in the purchase order line database table which 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 which holds the purchase order line item description.
DBPOQuantity	Freetext	Name of the column in the purchase order line database table which holds the order quantity for the purchase order line item.
DBUnitPrice	Freetext	Name of the column in the purchase order line database table which holds the unit price for the purchase order line item.
DBPOTotal	Freetext	Name of the column in the purchase order line database table which holds the overall total for the purchase order line item.
DBTaxCode	Freetext	Name of the column in the purchase order line database table which holds the tax code for the purchase order line item.
DBTaxJurCode	Freetext	Name of the column in the purchase order line database table which holds the tax jurisdiction code for the purchase order line item.
DBUOM	Freetext	Name of the column in the purchase order line database table which 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 which holds the purchase order line item price unit.
DBPUOM	Freetext	Name of the column in the purchase order line database table which holds the order price unit of measure for the purchase order line item.
DBTotalQuantity Delivered	Freetext	Name of the column in the purchase order line database table which holds the total quantity delivered to date for the purchase order line item.
DBTotalValue Delivered	Freetext	Name of the column in the purchase order line database table which holds the total value delivered to date for the purchase order line item.
DBTotalQuantity Invoiced	Freetext	Name of the column in the purchase order line database table which holds the total quantity invoiced to date for the purchase order line item.
DBTotalValue Invoiced	Freetext	Name of the column in the purchase order line database table which 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 which 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 which holds the

		charge code for the purchase order line item.
DBChargeCodeID	Freetext	Name of the column in the purchase order line database table which holds the charge code ID for the purchase order line item.
DBCompanyCode	Freetext	Name of the column in the purchase order line database table which holds the company code for the purchase order line item.
DBERPPOType	Freetext	Name of the column in the purchase order line database table which 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 which holds the PeopleSoft purchasing business unit for the purchase order line item.

4.1.27 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.

4.1.28 CTR Section

This section contains settings whereby the system can be pointed to a look-up 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 look-up 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.

4.1.29 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.

The available settings are described in the table below.

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 (for example, 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 (for example, 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.

4.1.30 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.

The following settings are configurable:

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 system 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 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 system 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 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 filetype) 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 (1234) and the vendor number (ABC) separated by a colon, the setting should contain: <%TNM>:<%INO>:<%VID> …which would produce the following: myFile.tif:1234:ABC Additional text can also be included in the format. For example: File=<%TNM>|Invoice Number=<%INO>|Vendor ID=<%VID> Would produce: File=myFile.tif|Invoice Number=1234|Vendor ID=ABC The columns are represented by the following symbols: ' <%TNM>= Name of the original imported file (no file path) ' <%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 (i.e. INVOICE or CREDIT) ' <%ITP>= Invoice type (i.e. PO or NO-PO) ' <%IDT>= Invoice date ' <%INO>= Invoice number ' <%TAX>= Invoice tax amount ' <%WTX>= Invoice withholding tax amount ' <%DCT>= Invoice discount amount ' <%MSC>= Invoice misc charge amount ' <%CUR>= Invoice currency ' <%TOT>= Invoice total ' <%PST>= PST/QST 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 Number (POR) ' <%PSN>= Payment Order Reference 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 ' <%ERP>= ERP document key ' <%EPT>= ERP system PO type ' <%BSU>= Business unit ' <%DEL>= Delivery note
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> The available literals are as follows: ' <%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>= Oracle Forms Recognition line item ' <%LPT>= Plant ' <%LTY>= ERP purchase order type ' <%LBU>= ERP purchasing business unit ' <%LCC>= Company Code ‘ <%LTR>= VAT / tax rate

4.1.32 SPC Section

The SPC section is where parameters to be used in the stored procedure look-ups 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 look-ups 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.
NN_ParameterType	Freetext	This is the stored procedure parameter type. Possible options are BOOLEAN, INT, DATE, DOUBLE and 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 here: 1) Specify the technical name of a Oracle Forms Recognition field (for example, PONumber, CompanyCode) – this is case-sensitive; 2) 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’.

5 CUSTOMIZATIONS AND USER EXITS

5.1 Introduction

The A/P Solution can be customized to befit client requirements. The customizations take 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 as described in Appendix E, changing or creating Verifier forms, or setting up new users.

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 on the 'UserExits' class script window. More information can be found on this subject in section 5.2.

5.2 Script Customizations

5.2.1 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.

5.2.1.1 Project Script Class

The project script class contains script associated with standard system events, which are known as the 'ScriptModule' events.

These 'ScriptModule' events are called at specific points within the Oracle Forms Recognition 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.

5.2.1.2 Global Variables Script Class

The global variables script class houses all global variables that are used within the A/P Solution. These data definitions are exposed so that a 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 on the user exit 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:

Name of function / sub-routine	**Description**
ReadPrjINI	This function reads the system configuration .ini file.
DicVal	This function returns the value of any parameter contained within the system configuration .ini file. The function parameters are as follows: strKey = the name of the .ini file parameter. strDic = the name of the .ini section in which the parameter is held. Neither strKey nor strDic are case-sensitive. Example usage: If .ini file parameter 'EXP_VL_OutputDateFormat' contains 'MMDDYYYY', then the following command will copy 'MMDDYYYY' into local string variable 'strOutputDateFormat': strOutputDateFormat = DicVal("OutputDateFormat", "EXP") For .ini file 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 an .ini file parameter value 'strVal' that has been entered against a parameter with Boolean type 'OP', and determines whether the value should be interpreted as 'YES' or 'NO'.
SplitString	This subroutine 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 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 date format used internally within Oracle Forms Recognition (i.e. DD/MM/YYYY) into a specified format. The interface parameters are as follows: strDate = date to be formatted strFormat = format of date (either 'MMDDYYYY', '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" ' 12th 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 Oracle Forms Recognition (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', 'MMDDYYYY' – any other entry will return 'DDMMYYYY' strSeparator = separator currently used in strDate Example usage: Dim myDate as String myDate = "2009-08-12" ' 12th August 2009 myDate = fnConvertToInternal(myDate, "YYYYMMDD", "-") The value of myDate will now be set to "12/08/2009".
fnFormatDateForExport	This function converts a date in the Verifier output format, as configured in the DAT section of the system configuration, into a date in the export output format, as configured in the EXP section. 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 (i.e. the contents of pField.Text or pWorkdoc.Fields("MyDate").Text). 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 f 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 .ini file parameter containing the tag for the XML field strValue = value of the field to be outputted Example usage: In the .ini file, 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' which 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 framework: 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 system 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 system configuration. The interface and function usage is identical to that of 'fnWriteXMLField' as 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 system 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 .ini file parameter containing the table column mapping for the destination field in the database strValue = field value to be written to the database strINIFileKeyName = current .ini file parameter being assessed by the database header output routine – this value should be passed unaltered from the value passed into the user exit 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 .ini file, a new parameter has been created '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' which 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 system 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 system configuration. The interface and function usage is identical to that of 'fnWriteDBHeaderField' as described above.
fnGetFileName	This simple function receives a full filename, which includes the file path and file extension, and returns the name of the file itself. For example, if 'c:\My Documents\12345.tif' is passed to the function, the output will be '12345'. Interface: strFileName
fnGetBaseClass	This simple function returns 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 child class of 'Invoices', then the function will return 'Invoices'. Interface: strClass
fnIsVerifier	This function returns a Boolean 'true' value if the current Oracle Forms Recognition Module executing the script is the Verifier Module.
fnGetBatchID	This function receives the path to a document in the batch directory (strWorkfile), 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 (upper or lower case). Interface: strString
fnGetUserDecimalSeparator	This function reads the Windows locale 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 strKey = CSV file group number (for example, 01, 02 etc..) strSymbol = user-defined literal to be replace (for example, <%ZIC>) strValue = value to replace the literal with Example usage: If CSV group '01' has the following 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 system 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 system configuration. The interface and function usage is identical to that of 'fnWriteCSVField' as 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 -1 and an appropriate error message will be written into the standard Oracle Forms Recognition logfile.
	
	Example usage:
	
	This following code will instantiate a database connection and execute an SQL call where variable 'myDBConnection' represent the connection string, and 'mySQL' represents the SQL statement (both string variables):
	
	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 ‘fnSetDBConnection’.
fnCheckDBArray	Utility function that checks to see 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 engine field ‘oASSA’ and document class ‘strClass’. Interface: strClass, oASSA
fnIsValueInList	This function takes a comma-separated list in input parameter 'strList' and a value ‘strValuePreserve’. The function will return a Boolean true’value if ‘strValuePreserve’ is one of the values in the list. Interface: strList, strValuePreserve
fnConvertToDouble	This utility function takes in a string ‘strString’ and converts it 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 utility function returns a Boolean true value if all characters passed in the input parameter ‘strTemp’ are numeric (i.e. 0-9). 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 system 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

5.2.1.3 User Exits 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.

For more detail on this subject, please refer to section 5.2.2.

5.2.1.4 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 code. Doing so may prove fatal to the running of the solution.

5.2.1.5 APPackaged / 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.

5.2.1.6 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 'play' one script if there is a dependency on a script which is not 'playing'. 'Playing' a script will also perform a syntax check.

For that reason, the scripts must be 'played' in sequence.

Within the project, the sequence is: GlobalVariables → UserExits → Project

5.2.2 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 'window' 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	Description and possible uses

UserExitCustomExport	ScriptModule_ ExportDocument	This is the user exit for custom export modules. For example, for custom flat files or custom database updates This is the only user exit which has a corresponding activation parameter within the EXP section of the system 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 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'. This user exit is described in greater detail in section 7.7.
UserExitPostExtract	Document_ PostEvaluate on the invoices class	This is the user exit used to set any custom field defaults, or to re-evaluate any extracted fields. Interface: pWorkdoc
UserExitRouteDocument	ScriptModule_ RouteDocument	This is the user exit for performing any custom activity connected to the 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
UserExitPONumber PostEvaluate	PONumber_ PostEvaluate on the 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
UserExitVoidDocument Export	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.
UserExitPONumber Validate	PONumber_Validate on the 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.
UserExitPONumber ValidateStart	PONumber_Validate on the 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.
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
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
UserExitDocumentType Validate	DocumentType_Validate on the invoices class	This user exit is called at the beginning of DocumentType_Validate on the invoices class. The exit 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
UserExitAmountMisc PostEvaluate	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 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 Oracle 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 A/P project.. Standard configuration permits different tolerance values to be assigned to different currencies, but if a further dimension is required (for example, to consider the company code), then this logic should be implemented within this user exit. Adjusting the tolerances requires manipulating the 'Tolerance' array, which consists of the following properties: 'HeaderTolerance' - tolerance to apply to header amount fields 'TableRowTolerance' - tolerance to apply to table rows 'TaxTolerance' - tolerance to apply to the validation of the tax amount during the automatic tax code validation 'NoDecimalPlaces' - Boolean to indicate whether decimal places are expected for the invoice currency Interface: Tolerance, pWorkdoc
UserExitDocumentOnAction	Document_OnAction on the 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 Verifier Design Mode. 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 .ini file parameter (EXP_VL_DBHCxxx) must be added to the system 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 ini file parameter For example, if the ini file 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 file path of a URL, depending on settings in the REP section of the system 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: 1) The document attributes (for example, class name, scan date) 2) The invoice header (for example, invoice number, date) 3) The line item detail (for example, quantity, unit price) 4) General Ledger Coding (for example, 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 group number for the file being processed. Interface: pWorkdoc, strRecordText, strKey An example illustrating how this user exit may be used can be found in section 7.6.2.
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 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 An example of how to use this user exit can be found in section 7.6.3.
UserExitExportSuccess	ScriptModule_ ExportDocument	This user exit is called at the point where it is known that export is 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 post evaluate 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. The interface of the user exit is as follows: pWorkdoc PO() – this array contains the details of the purchase orders to be used during line pairing. It is based on the POKey structure, described in section 5.2.5.2. 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_Verifier Exception	This user exit is triggered when a user send a document to an exception state in Verifier. Interface: pWorkdoc, Reason, CreateNewBatch, BatchName, BatchDocumentState, BatchPriority, BatchFolderName, ApplyExceptionHandling Details on this interface can be found in the Oracle Forms Recognition Scripting Guide.
UserExitFilter VendorSearch	Document_Post Extract, internal application	This user exit allows a developer to filter the list of vendors shown in the Verifier vendor search facility, and also to 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.
		
		The interface is as follows:
		
		pWorkdoc – Oracle Forms Recognition workdoc object
		oCandidate – vendor candidate object
		Address – vendor address structure (see section 5.2.5.4)
		strDisplay – current vendor search box display line for 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.
		blReject – this 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.
		
		Example usage 1:
		
		In this example, the client wishes to exclude any vendors from being extracted automatically and from appearing for selection in the vendor search if they are not set up for the invoice company code.
		
		This can be achieved through the following single line of code:
		
		`If Not fnIsValueInList(Address.CompanyCodes, pWorkdoc.Fields("CompanyCode").Text) Then blReject = True`
		
		This assumes that a comma-separated list of valid company codes for which the vendor is set up for is available in the vendor master extract, and the correct column is mapped against the 'CompanyCodes' parameter in the SRC section of the system configuration.
		
		Example usage 2:
		
		The clients wishes to exclude any vendors from being extracted automatically if they are not set up for the invoice company code, but does wish to allow the Verifier user to select those vendors via a vendor search.
		
		This can be achieved via the following single line of code
		
		`If strDisplay = "" And Not fnIsValueInList(Address.CompanyCodes, pWorkdoc.Fields("CompanyCode").Text) Then blReject = True`
		
		The same assumptions for example 1 apply.
		
		Example usage 3:
		
		The client wishes to lower the confidence of any vendor candidates that do not belong to the invoice company code by 30%.
		
		This can be achieved by the following single line of code:
		
		`If strDisplay = "" And Not fnIsValueInList(Address.CompanyCodes, pWorkdoc.Fields("CompanyCode").Text) Then oCandidate.Weight = oCandidate.Weight - 0.3`

		Example usage 4: The following single line of code will adjust the display in the vendor search box so that only the vendor name, vendor street address and zip code are displayed: `If strDisplay <> "" Then strDisplay = Address.Name & ", " & Address.Address & ", " & Address.Zip` Interface: pWorkdoc, oCandidate, Address, strDisplay, blReject
UserExitSetReporting LoginName	Internal application	This user exit allows a developer to change the name of the user used in the reporting tables. 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). It can be used to skip the MIRA process of line pairing by setting dblPOTotal and dblGRTotal both 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 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' array (for structure, see section 5.2.5.4). The user exit will not be called if the vendor details have already been read and loaded into the local cache. **Example Usage:** The client has been unable to supply the vendor country of origin as a standard two-character ISO-code, as required by the application. The client is only doing business with vendors from the US and Canada, and is able to provide 'USA' and 'CAN' as the

		country codes.
		
		The following script in the user exit will convert the client's country code to the required application country code:
		
		`Select Case Address.Country`
		`  Case "USA"`
		`  Address.Country = "US"`
		`  Case "CAN"`
		`  Address.Country = "CA"`
		`End Select`
		
		Interface: oASSA, strVendorID, Address

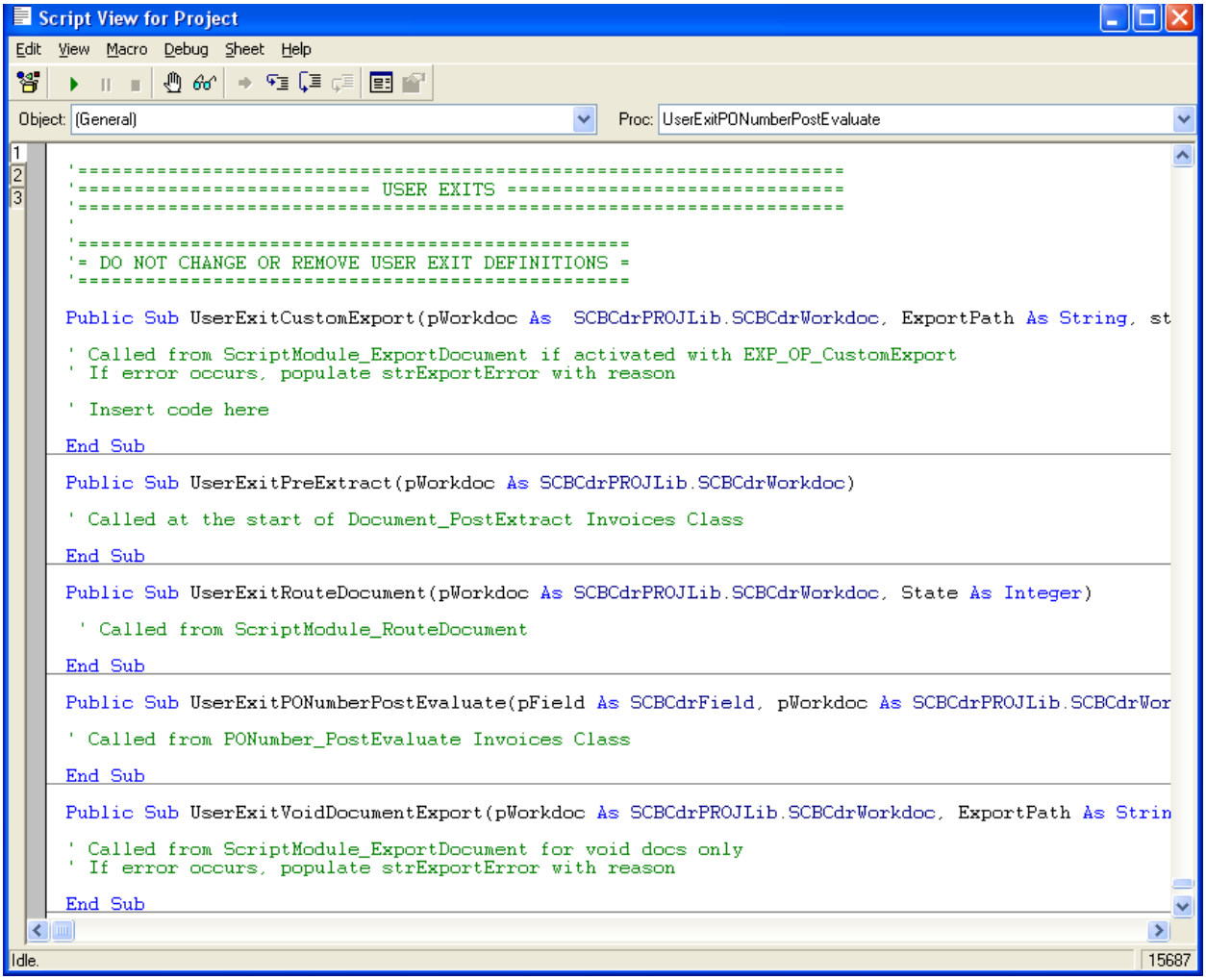
UserExitDocument Validate	Document_Validate on the Invoices class script level	This user exit is called from 'Document_Validate' on the 'Invoices' class script level. It can be used to code in additional document level validations and activities. Interface: pWorkdoc, pValid

UserExitInvoiceNumber Validate	InvoiceNumber_ Validate on the 'Invoices' class	This user exit is available to include additional validations and formatting against the invoice number field.
		
		Example usage:
		
		The client's business rule is that, if an invoice number is extracted that is longer than 16 characters, the system should format the invoice number so that only the last 16 characters are retained. The following single line of code will accomplish this:
		
		`If Len(pField.Text) > 16 Then`
		`pField.Worktext.Text = Right(pField.Text, 16)`
		
		Interface: pField, pWorkdoc, pValid

If the required customization requires a user exit that is not present, then a user exit enhancement request should be made to Oracle.

5.2.3 Sample User Exit Screenshot

The screenshot below illustrates the UserExits class script:



5.2.4 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 system 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 system configuration, the following parameter has been added:

ERR_VL_900=Please check data entry

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

Dim strError As String

strError = DicVal(“900”, “ERR”)

Hence, local string ‘strError’ will now contain ‘Please check data entry”.

5.2.5 Project Data Structures

Oracle Forms Recognition 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:

5.2.5.1 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 based 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
UserExitDolphinWorkflow
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	'X' or blank	Subsequent debit/credit indicator This denotes whether the line item is a subsequent debit of 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 (i.e. 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, as used in countries that have tax jurisdictions for their sales tax (for example, the US, Canada, Brazil). It is populated only if the line item is paired.
COND_TYPE	String	Condition type (not applicable)
COND_ST_NO	String	Condition step number (not applicable)
COND_COUNT	String	Condition count (not applicable)
FREIGHT_VEN	String	Freight vendor ID (not applicable)
REF_DOC	String	Reference document number (not applicable)
REF_DOC_YR	String	Year of reference document number (not applicable)
REF_DOC_IT	String	Item in reference document number (not applicable)
SHEET_NO	String	Service entry sheet number (not applicable)
SHEET_ITEM	String	Service entry sheet item number (not applicable)
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 read 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' and "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 read 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 lifted from the purchase order line item to which an invoice line is paired.
CHARGECODE	String	Invoice line item charge code This is lifted from the purchase order line item to which an invoice line is paired.
CHARGECODEID	String	Invoice line item charge code ID This is lifted from the purchase order line item to which an invoice line is paired.
MATERIALGROUP	String	Material group This is lifted from the purchase order line item to which an invoice line is paired.
DISTILLER_LINE	String	Original line item number This is the original line item number in the 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 lifted 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 lifted 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.
MISCCHARGE	'X' or blank	'X' indicates that this is miscellaneous charge line item. The flag will be blank if the system has identified the line as a non misc-charge item.

5.2.5.2 POKey Structure

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

The structure is used in 'UserExitLinePairingPOs', and is defined 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 (for example, 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.

5.2.5.3 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.

It is used in the interfaces of the following user exits:

```
UserExitCustomExport
UserExitPostLinePairing
```

The structure consists of the following components:

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

5.2.5.4 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 system configuration.

It is used in the interfaces to the following user exits:

```
UserExitCustomExport
UserExitFilterVendorSearch
UserExitAddressArray
```

The structure consists of the following components:

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 For US addresses, the state code would be expected here (i.e. CA = California, VA = Virginia etc…)
COUNTRY	String	Vendor country This should be the two character ISO-code for the country (i.e. US = United States Of America, DE = Germany, GB = United Kingdom etc…)
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 or not
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 which 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 system 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: BankAccount,SortCode,ERPBankAccountCode A sortcode is the US equivalent of a routing number.
WHTAXDETAILS	String	Vendor withholding tax details (not applicable)
UTILITYFLAG	String	Indicator as to whether the
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
ALTERNATEPAYEE	String	ERP system alternate payee details (not applicable)
PERMITTEDPAYEE	String	ERP system permitted payee details (not applicable)
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.

5.2.5.5 Tolerance Structure

The tolerance structure holds a series of limits the system uses when performing mathematical validations on the amounts read from the invoice. The tolerance limits are set by the tolerance group assigned to the invoice currency.

The Tolerance structure is used in the interface for the following user exit:

`UserExitSetTolerance`

The structure consists of the following components:

Structure element	Type	Description
HEADERTOLERANCE	Double	This is the maximum amount by which the document header amounts (i.e. total = tax + freight + sum of line items/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 (i.e. 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 read 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 (for example, pennies, cents). Common world currencies that do not have sub-units are the Hungarian Forint and the Japanese Yen.

5.2.5.6 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.

It is used in the interface of the following user exit:

`UserExitCustomExport`

The structure consists of the following components:

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 system configuration determine whether 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 the 'VENDOR NOT FOUND' or 'MISSING/INVALID VENDOR & PO' invalid reasons in Verifier.
NOPO	Boolean	This flag is set to 'true' if the user has selected either the 'MISSING/INVALID PO' or the 'MISSING/INVALID VENDOR & PO' invalid reason 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 NO-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 hold: 1) Line item extraction is switched off for the project;

		2) An invalid reason of 'MISSING/INVALID VENDOR & PO' is set; 3) An invalid reason of 'MISSING/INVALID PO' is set and line items are not required under such circumstances; 4) An invalid reason of 'VENDOR NOT FOUND' is set and line items are not required under such circumstances; 5) The vendor has been identified as a utility vendor and line items are not required for utility vendors.

5.2.5.7 AccountingData Structure

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

It is used in the interfaces of the following user exits:

```
UserExitCustomExport
UserExitXMLOutput
```

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

The structure consists of the following components:

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
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	Asset sub number
NETWORK	String	Project network
ACTIVITY	String	Project activity
RL_EST_KEY	String	Internal key for real estate object
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 (above)
PO_PR_UOM	Double	Order price unit of measure
PERCENT	Double	Distribution percentage

5.2.6 Triggering of User Exits in Verifier

User exits are triggered when a user is working a problem document in Verifier. The following table lists the user exits that will be fired when a user performs a certain task in the order in which they are fired:

Verifier action	User exits
User hits enter on the document type field	UserExitDocumentTypeValidate
User hits enter on the invoice type field	**Always called:** UserExitPOValidateStart **Potentially called:** UserExitPOValidate (only if the PO is validated successfully against a database) UserExitAddressArray (if a vendor has not been loaded into the buffer)
User hits enter on the invalid reason field	**Always called:** UserExitPOValidateStart **Potentially called:** UserExitPOValidate (only if the PO is validated successfully against a database) UserExitAddressArray (if a vendor has not been loaded into the buffer)
User hits enter on the invoice number field	UserExitInvoiceNumberValidate
User hits enter on the vendor ID field	**Always called:** UserExitPOValidateStart **Potentially called:** UserExitPOValidate (only if the PO is validated successfully against a database) UserExitAddressArray (if a vendor has not been loaded into the buffer)
User hits enter on the purchase order number or purchase order number extension fields	**Always called:** UserExitPOValidateStart **Potentially called:** UserExitPOValidate (only 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 box, 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 **Potentially called:** UserExitSetReportingLoginName (if Visibility reporting is activated)

6 BASIC INSTALLATION GUIDE

6.1 Introduction

This section describes the steps required to carry out a basic installation of the A/P project.

The topics covered in this section are:

1) Creating the Oracle Forms Recognition folder structure
2) Installing the Oracle Forms Recognition A/P Project File
3) Setting up the vendor extract file and pool
4) Performing basic system configuration settings
5) Configuring a standard output file
6) Setting up the Runtime Server instances
7) Setting up the Verifier application
8) Processing documents through the system

Upon completion of the steps above, the necessary pre-requisites will be in the place to process unseen invoice documents through the system, extract invoice data along with an effective means to correct problem documents and to provide a basic output.

Before embarking upon these steps, the Oracle Forms Recognition developer should ensure that:

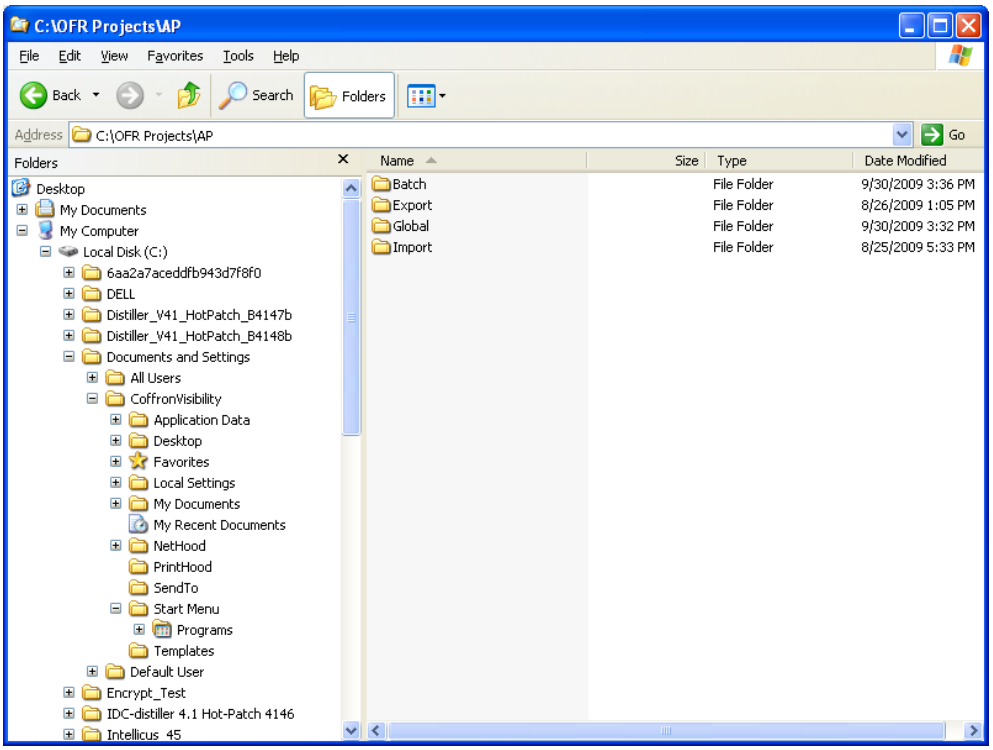
1) The software installation of Oracle Forms Recognition has taken place to the minimum supported version (10.1.3.5.0, Build 4161).
2) A vendor extract file / database table has already been made available by the client or manually created.

6.2 Creating the Oracle Forms Recognition Folder Structure

Following a successful installation of the software, the first step to install the A/P project is to create the underlying folder structure within Windows Explorer.

In the example below, the folder structure is going to be created on the C drive in a folder called ‘Oracle Forms Recognition’, sub-folder ‘AP’.

Within this directory, four folders should be created: Import, Global, Batch and Export. This is shown in the screenshot below:

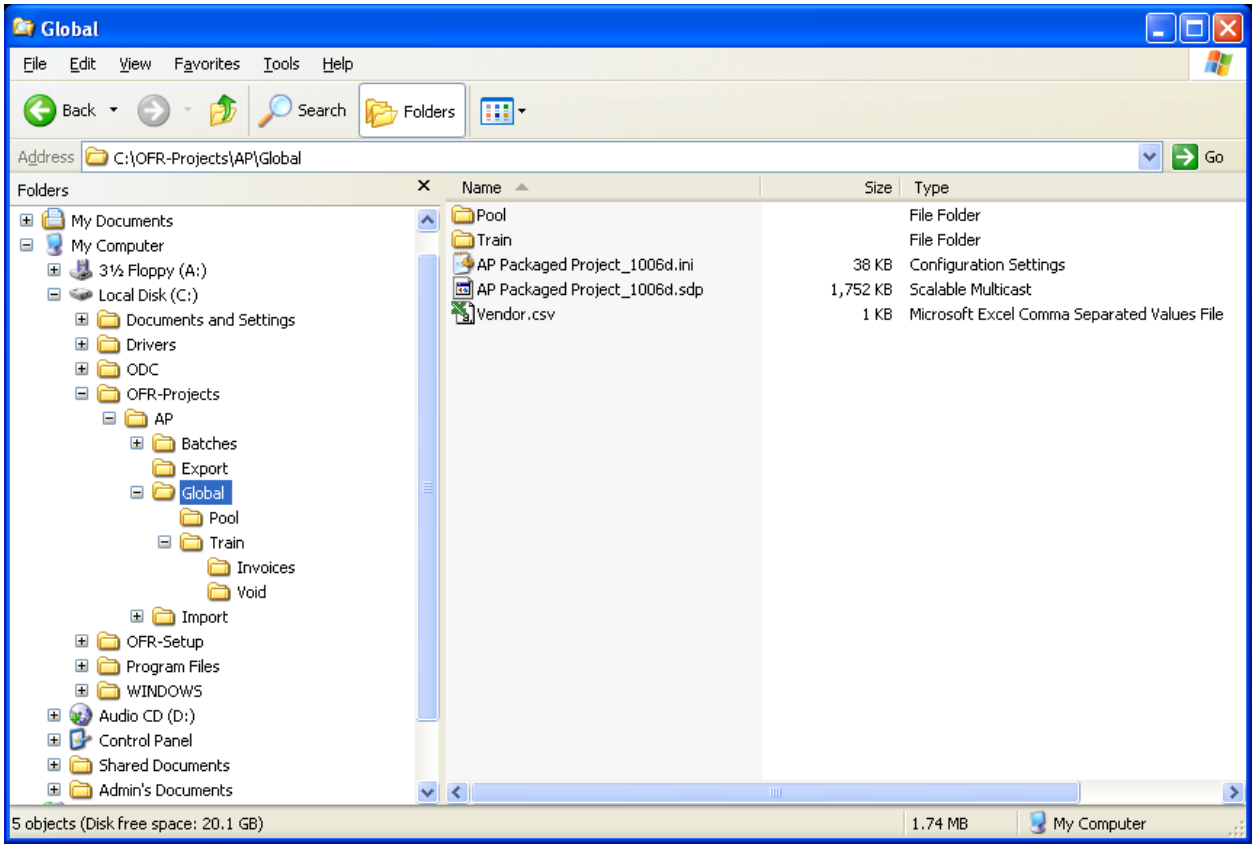


6.3 Installing the Oracle Forms Recognition A/P Project File

Open the newly-created ‘Global’ directory, and copy in the project components. These components include:

1) The project file (.sdp)
2) The train directory including all contents
3) The project .ini file

Once complete, the global directory should appear as follows:



It is recommended to not change the project file name since the provided name includes the version number of the project. If you do change the name of the project file, the project ini must also be renamed to reflect the same base file name.

6.3.1 Logging in to the A/P Project

When using the A/P Project for the first time, use the default value **Administrator** for the user ID and a blank password. For information about changing these default values after initial login, see the section on creating user accounts and groups in the Designer User Guide.

6.4 Setting up the Vendor Extract File and Pool

The next step is to install the vendor extract file and set up the pool directory used by the Oracle Forms Recognition Associative Search engine.

The Associative Search Engine is a patented technology that identifies 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 that it is tolerant of OCR problems on the document, difference between the way that vendor details are expressed on the invoice and how they are represented in the vendor master data, and vendor names embedded in picture logos.

The vendor master extract must be made available, which can be delivered within a database table, which the system will connect to via ODBC, or in an ASCII Text file.

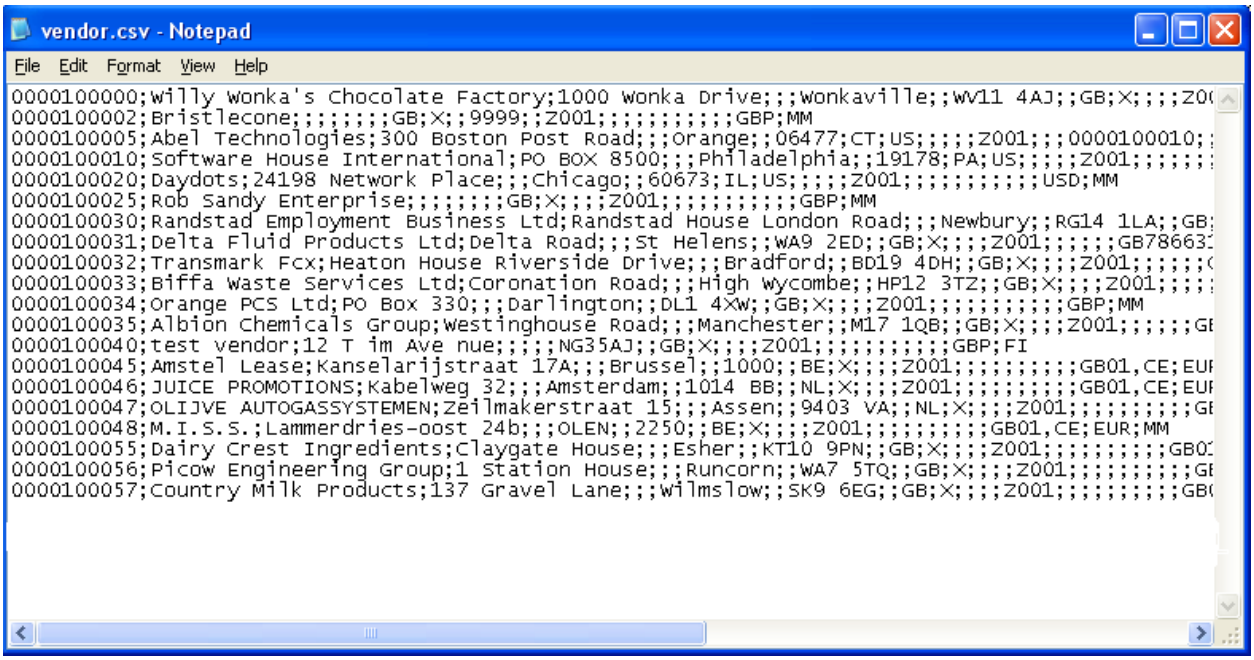
If an ASCII Text file is provided, then it must be compliant with the following:

1) Each row in the file should represent a single vendor at a single address.
2) 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.
3) Each row in the file must have one column that is a unique identifier for that record and is common only to that row.
4) Each row in the file must have an equal number of columns.
5) 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.
6) Each column should be stripped of double-quotes (“).

It is essential to have a column which represents the vendor country included within the extract. This should always be the two character ISO-code for that country:

E.g. United States = US, United Kingdom = GB; Germany = DE; Switzerland = CH etc.

A screenshot from a sample vendor extract file can be found below:



If the ERP vendor master data includes intercompany vendors, these should be excluded from the file, as should employees who are set up as vendors. Intercompany vendors are handled in

a different way within the Oracle Forms Recognition A/P Solution. More information on this topic can be found in Appendix H: Handling Intercompany Vendors.

Full step-by-step instructions on how to set up the pool and the associated configuration required can be found in Appendix D: Configuring the Vendor ID Field.

Finally, for the purposes of a basic installation, disable the employee and company code ASSA fields as described in Appendix G: Deactivating ASSA Fields.

6.5 Performing Basic System Configuration Settings

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

The following areas are covered:

1) Activating reporting to a reporting schema
2) Configuring purchase order number formats
3) Configuring bill-to name formats
4) Configuring tax rates

6.5.1 Activating Reporting to the Reporting Database

The settings required to activate solution reporting so that the progress of documents as they flow through the system is captured in a reporting schema can be found in the REP and SQL sections of the system configuration.

Reporting can be switched on 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 an Oracle reporting database.

SQL connection strings are set in the SQL section of the system configuration. Within this section, each connection string is specified and assigned to an SQL connection group.

The Oracle Forms Recognition A/P Solution supports multiple database connection strings as the reporting database may not be the same database that is used to validate a purchase order number against, which, in turn, may not be the same database that data is exported to. Each database connection string is assigned a group number which 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 a SQL Server connection string for connection group '01'. Groups '02' and '03' are as of yet undefined. Additional groups can be added as required.

```
SQL_VL_01_ConnectionString=Provider=SQLOLEDB.1;Password=test;Persist Security
Info=True;User ID=test;Initial Catalog=Invoices;Data Source=WXP-ORACLE
\SQLEXPRESS,1254
SQL_VL_02_ConnectionString=
SQL_VL_03_ConnectionString=
```

In the REP section, this connection group is specified against the 'SQLConnectionGroup' parameter so that the system knows which connection string to use for the reporting database:

```
REP_OP_ConnectToReportingDB=YES
REP_VL_SQLConnectionGroup=01
```

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.

Oracle 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 is 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. So, if the image filename is '12345.tif', then key for this record in the reporting database will be set to '12345'. If Oracle 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 database record key as long as it is bound by an underscore. To do this, two parameters must be set as illustrated in the example below.

Example:

The filename provided by the upstream software is ABC_12345_20091007.tif.

The unique document identifier is the second component of the filename (i.e. '12345'), and is the handle on the database record in the reporting table.

In the IMP section of the system configuration, the following parameter should be set:

```
IMP_VL_URN=COMPONENT2
```

In the REP section of the system configuration, the following parameter should also be set:

```
REP_VL_ReportingKey=URN
```

Some DB Administrators may insist on a specific naming convention for any tables created within their own databases, the names of the standard reporting tables are configurable via the following parameters:

```
REP_VL_ReportingDBDocumentTable=OFRdocument
REP_VL_ReportingDBFieldTable=OFRfields
```

```
REP_VL_ReportingDBHistoryTable=OFRdocstatus
```

The standard table names are shown in the example above; these table names are not case-sensitive.

Information is not written to the reporting database from any documents processed via the Oracle Forms Recognition Designer module, though this can be activated for testing and debugging purposes by setting the following parameter to 'YES':

```
REP_OP_ReportingInDesigner=YES
```

Typically, in production systems, this should always be set to 'NO'.

6.5.2 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 system configuration. If no configuration or 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 via a simple expression, which uses wildcard characters for unknown digits or letters. A '#' sign represents a number; a '@' sign represents a letter; and a '?' sign represents any number, letter or special character. The format is not case-sensitive.

Hence, to configure a purchase order number that is always 7 digits in length beginning with '10', an appropriate format would be:

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

Equally, if an additional purchase order number format is used which is 8 characters in length, two alpha and 6 numeric where the number always begins with '14', then 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.

For example:

```
PON_VL_01_Format=10#####
PON_VL_01_Ignore=-

PON_VL_02_Format=@@14####
PON_VL_02_Ignore=-
```

The above would permit a hyphen in any position in a purchase order number matching the formats 10##### and @@14####, hence 10-12345, 1-012345, 101234-5, AB-141234, A-B141234 or 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 – '01', '02' etc.

If the same parameter is found more than once then the system will fail to load the system configuration correctly and documents will throw errors during processing.

For example:

```
PON_VL_01_Format=10#####
PON_VL_01_Ignore=-

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

6.5.3 Configuring Bill-To Name Formats

The bill-to name is an optional field within Oracle Forms Recognition, 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, then it is necessary to configure key-words for the bill-to name so that the system is able to identify it, and extract it correctly.

Valid key-words are configured in the BTO section of the system configuration in a similar fashion to purchase order number formats (see section 6.5.2). 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=.,
```

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

6.5.4 Configuring Tax Rates

This is not a required activity, but it is strongly recommended for project implementations that involve processing invoices from countries where VAT/Sales Tax is set at the national level. For such countries the 'PrimaryTaxRates' and 'SecondaryTaxRates' parameters in the TAX section of the system configuration should be populated. Doing so will increase the success of the 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 activity 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/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' rate refers to the standard rate of tax applied to purchases in those countries (for example, 15% or 17.5% in the UK, 19% in the Netherlands and Germany, or 19.6% in France). A 'secondary' 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, 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=17.5,15,19,19.6
TAX_VL_SecondaryRates=5,6,7,5.5,2.1
```

6.6 Configuring a Standard Output File

The output of the Oracle Forms Recognition A/P Solution is governed by the settings in the EXP section of the system configuration (see section 4.1.25).

Within this section, there are configuration options to output data in a variety of configurable flat file 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 available date formats are 'YYYYMMDD', 'MMDDYYYY' and 'DDMMYYYY'. If this parameter is left blank, or a format not in that list is specified, the system will default to an output format of 'DDMMYYYY'.

If is also possible to configure a separator – for example, a full stop/period (.), a hyphen (-) or a forward slash (/). Single (') or double quotes ("), greater than (>) or less than (<) symbols are not permitted as separators, and, if specified, will not be visible in the output.

Example:

An invoice date is stated on the document as 2nd November 2008.

The configuration is set as follows:

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

The corresponding output date will be 11/02/2008.

A screenshot of an example output file can be found in Appendix G.

6.7 Setting up the Runtime Server Instances

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

To open the administrational console, the following path should be selected:

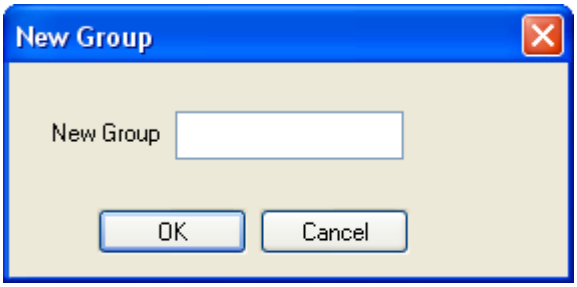
Start→ Programs→ Oracle Forms Recognition→ Runtime Service → Management Console

If the RTS service has not been started, then the following will start the service and open the administrational console:

Start→ Programs→ Oracle Forms Recognition→ Runtime Service → Start Runtime Service

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

To do this, right-click on the 'Runtime Server Administration' node and select 'New RTS Group…'. Enter the name of the RTS Group in the pop-up dialog box, for example:



Once the group has been created, it is now possible to add individual machines to the RTS group. Oracle Forms Recognition employs a scalable server architecture which permits the processor power of multiple machines in order 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.

For the purposes of a basic installation, it is assumed that the Oracle Forms Recognition environment will consist of a single master server machine. To add this machine to the newly created RTS group, right-click on the RTS group and select 'New Machine…' from the context menu.

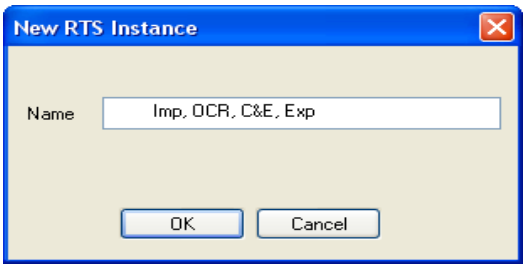
In the dialog box, select the domain of the machine to be added from the drop down list (or leave blank/leave as the default if the local machine is to be added), and enter the technical name of the computer in the 'Available Oracle Forms Recognition machines' field. Hit enter to confirm. The dialog box will now disappear, and the local machine will appear as a sub-node to the RTS group. As this installation covers the addition of a single master server to the RTS group, no more machines will be added.

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

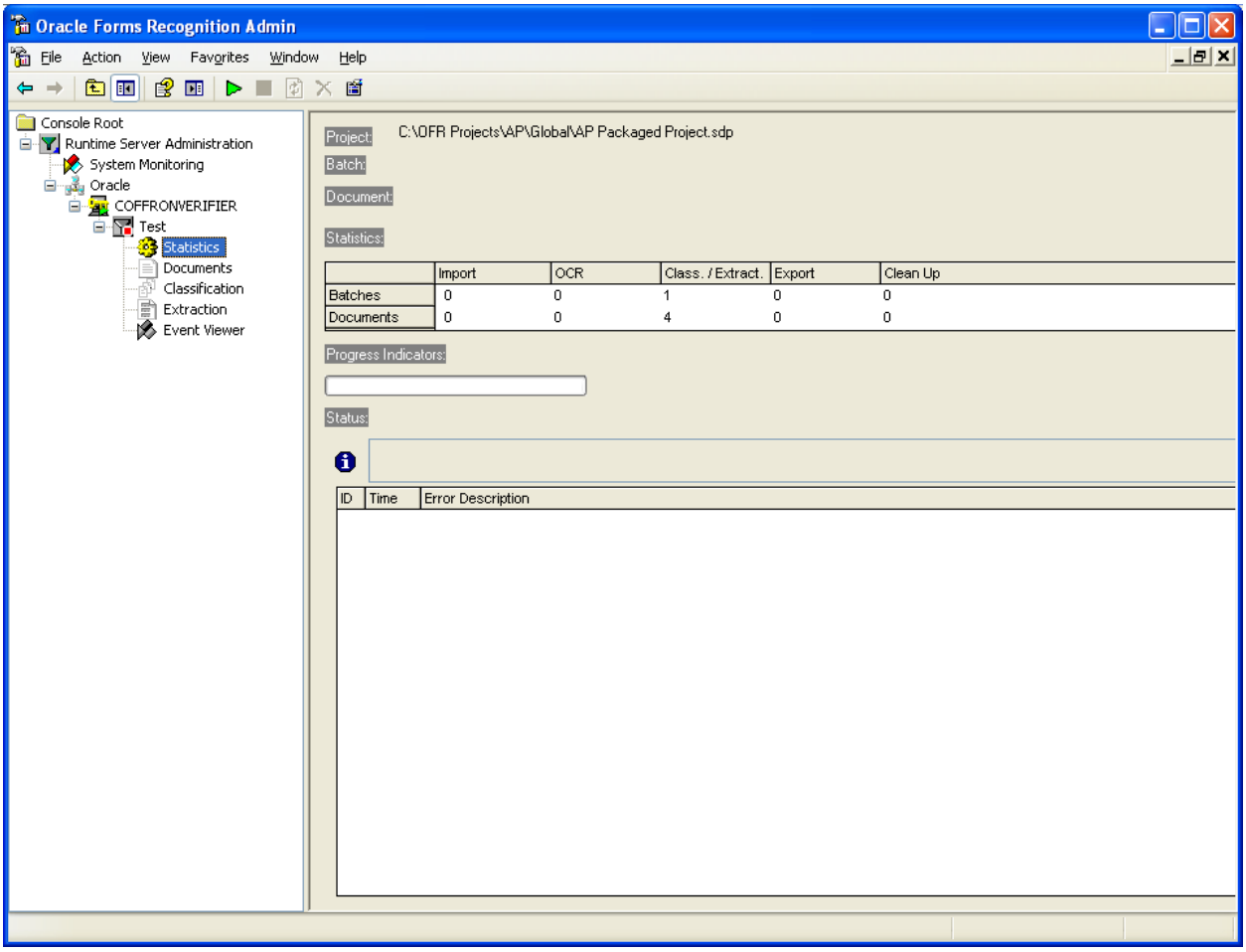
A single machine may have more than one RTS instance created for it, each instance carrying out different steps in the workflow. It is recommended that the number of instances carrying out OCR, classification and extraction, which are highly resource-intensive, do not exceed the number of available processors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

In the following example, just one instance is to be used, which will carry out Import, OCR, Classification, Extraction and Export.

To create an instance, right click on the machine name and select 'New' → 'RTS Instance…' from the context menu. Enter the name of the instance in the dialog box:



It is common practice to make reference to workflow steps that the instance carries out in the instance name. The screenshot below shows the status of the Runtime Server Administration console thus far:



To set the properties of the instance, right click on the instance name and select ‘Properties…’ from the context menu. Fill out the ‘General’ tab of the properties as follows, paying due respect to the file paths and directories that were established in the steps of sections 6.2 and 6.3. The critical file paths to be populated are the path to the project file, the path to the batch directory and the path to the export directory.

In a production environment, file paths should always be expressed as a UNC path, and, hence, the equivalent of the ‘OFR Projects’ directory should have a sufficient level of share.

The ‘Wait’ property denotes the time interval between scans of the batch and import directories; the ‘Enable batch integrity’ flag should only be checked for one RTS instance per batch directory.

i.e. If four RTS instances are all pointing at the same batch directory, then the ‘Enable batch integrity’ flag need only be checked for one of those instances.

On the ‘Workflow’ tab, select the Oracle Forms Recognition workflow steps that the instance is to perform. For this example, this single instance will carry out all steps aside from ‘Clean Up’.

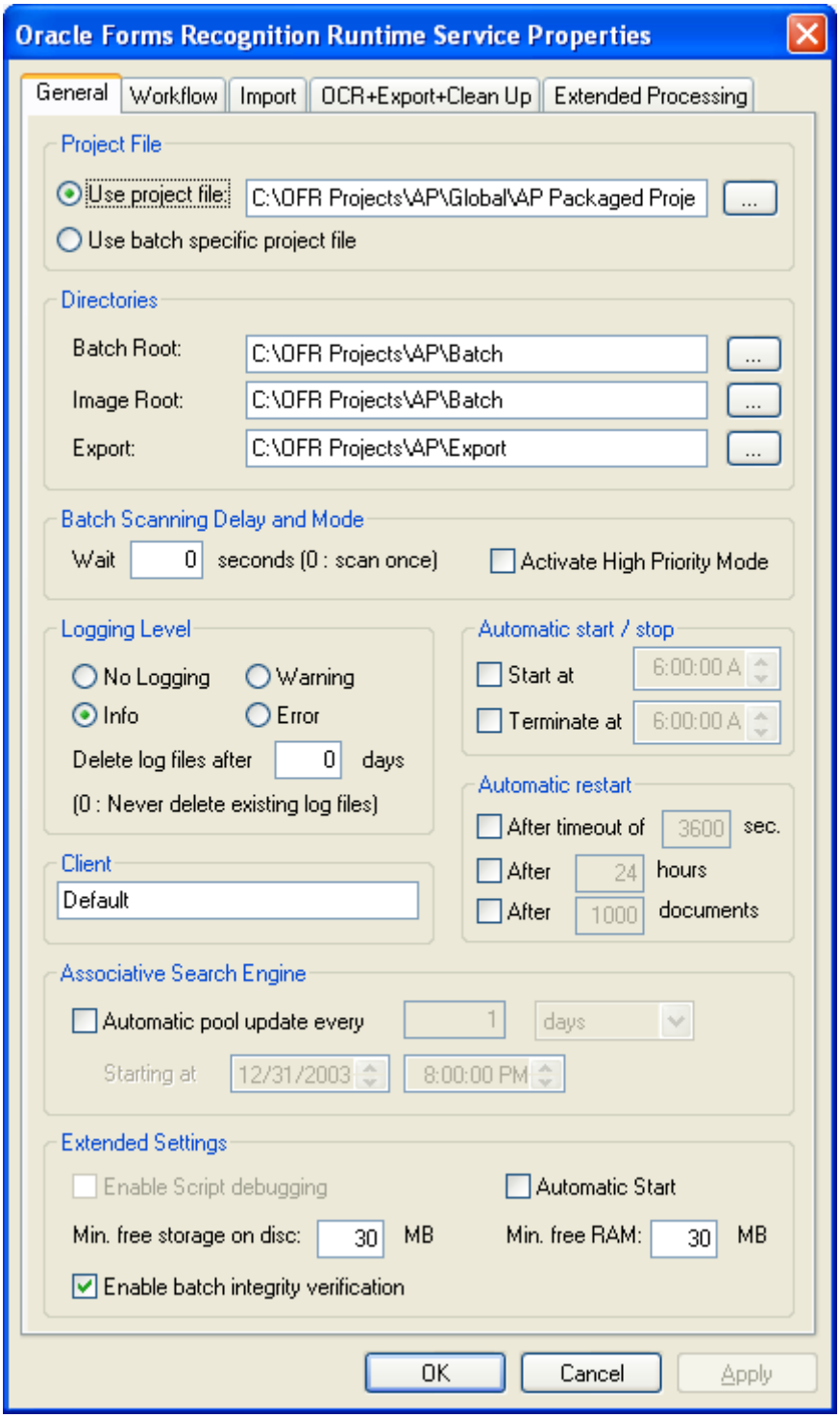
On the ‘Import’ tab, enter the path to the import directory, and ensure that the two radio button groups are set to ‘1 folder per batch’ and ‘Always import documents’. If the file types to be

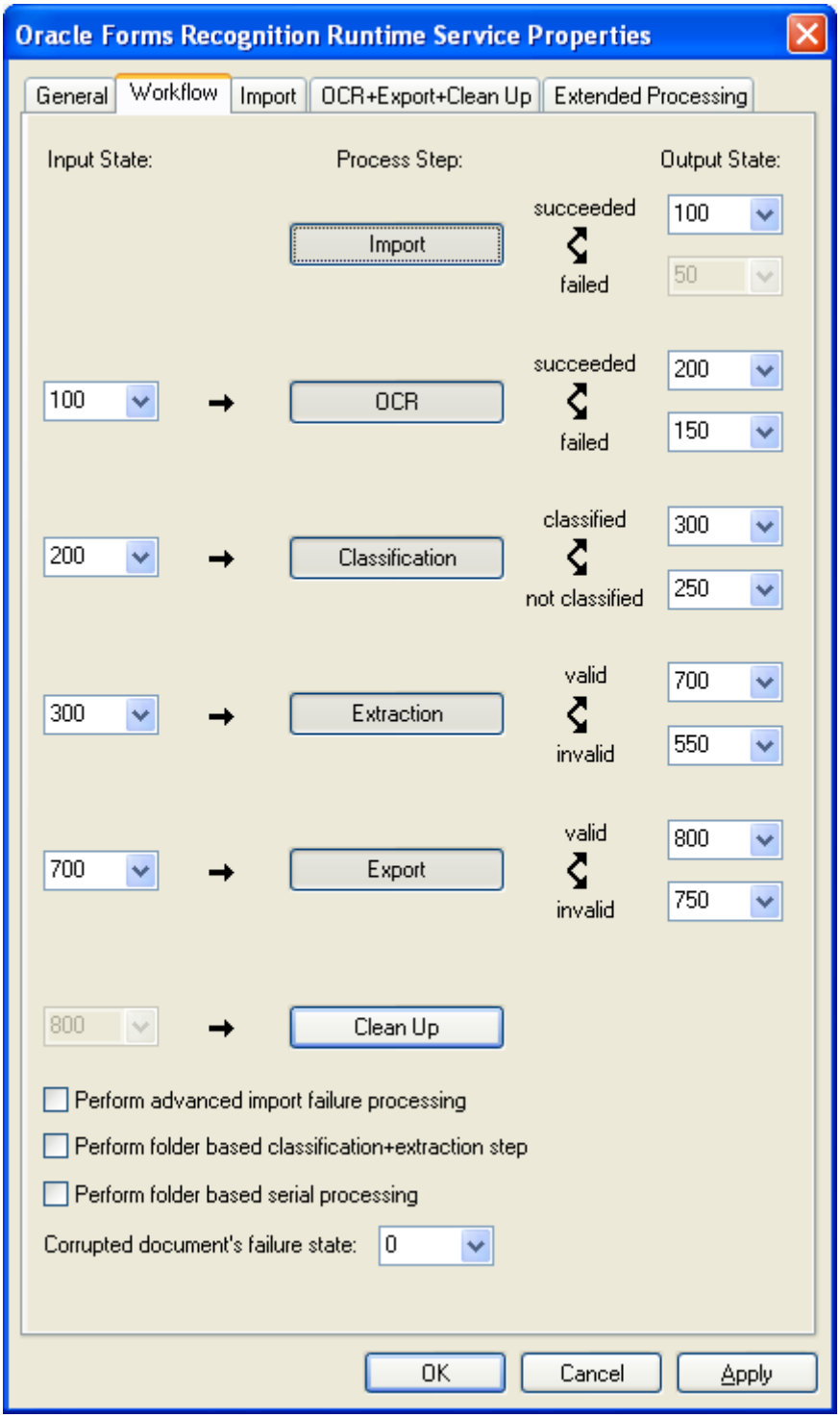
imported are tiff files, then the document type section can be left as the default values. For other document types, the ‘automatic’ checkbox should be checked.

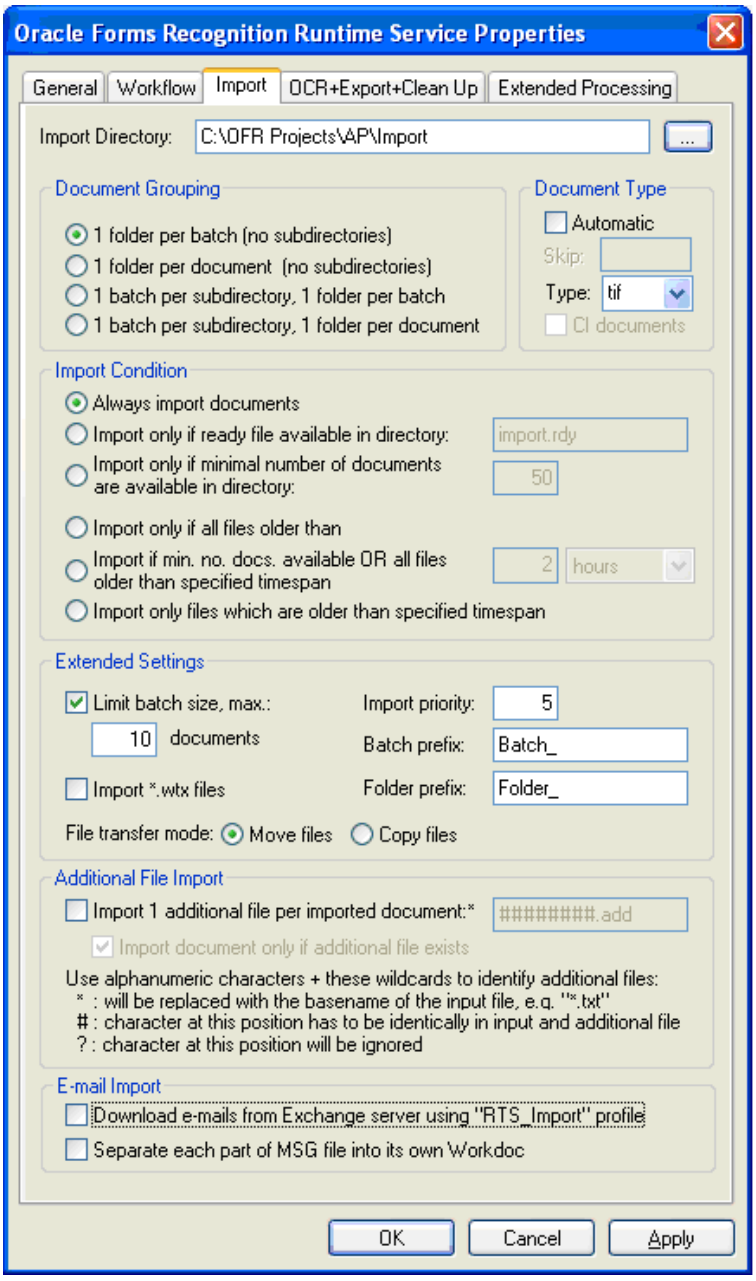
There is also an opportunity here to limit the number of documents in a single batch.

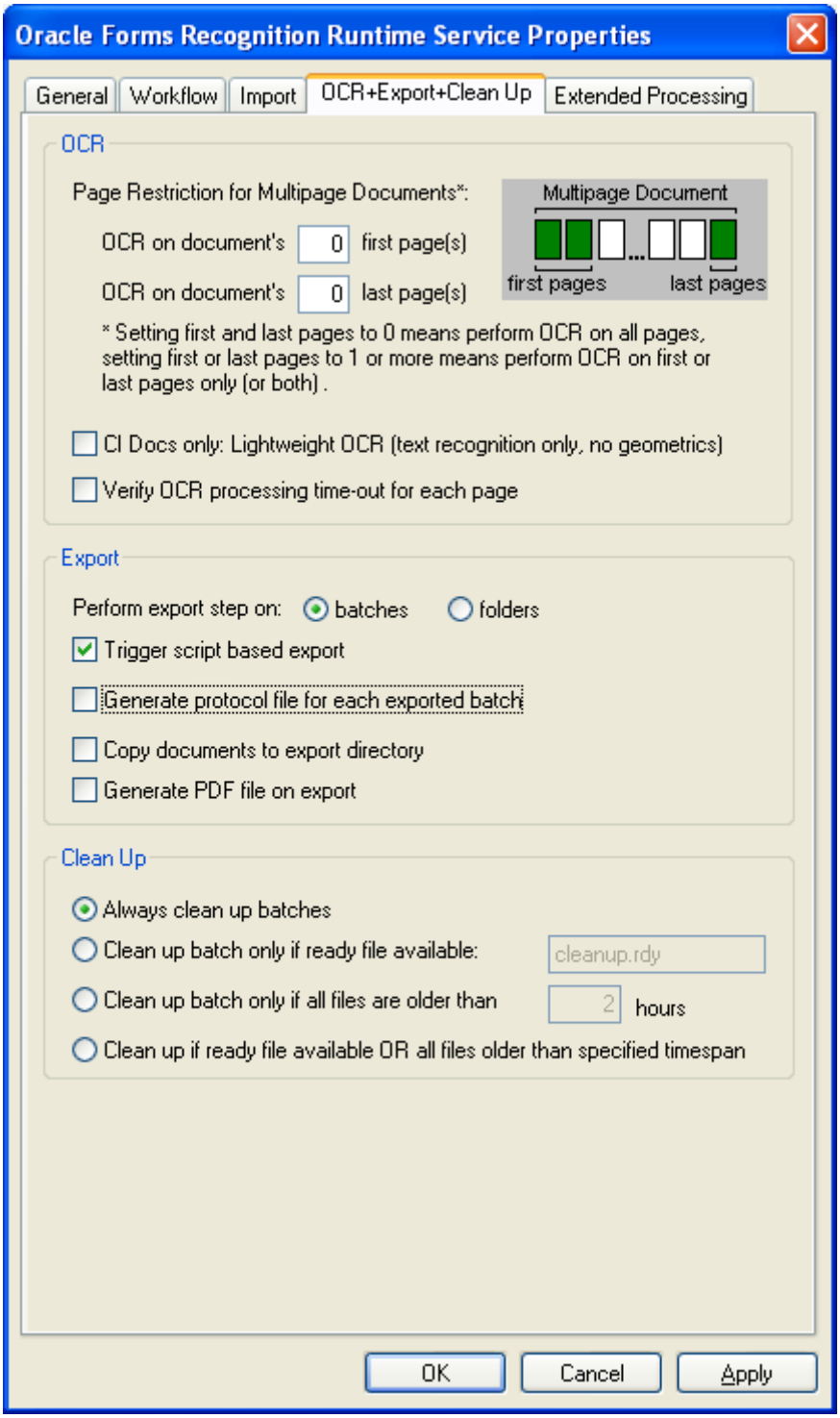
On the ‘OCR+Export+Clean Up’ tab, ensure that only the ‘Trigger script based export’ option is checked in the ‘Export’ section.

The screenshots below illustrate sample configurations:


Oracle Forms Recognition Runtime Service Properties
General
Workflow
Import
OCR+Export+Clean Up
Extended Processing
Project File
Use project file:
C:\OFR Projects\AP\Global\AP Packaged Proje
...
Use batch specific project file
Directories
Batch Root:
C:\OFR Projects\AP\Batch
Image Root:
C:\OFR Projects\AP\Batch
Export:
C:\OFR Projects\AP\Export
Batch Scanning Delay and Mode
Wait
0
seconds (0 : scan once)
Activate High Priority Mode
Logging Level
No Logging
Warning
Info
Error
Delete log files after
0
days
(0 : Never delete existing log files)
Automatic start / stop
Start at
6:00:00 A
Terminate at
6:00:00 A
Automatic restart
After timeout of
3600
sec.
After
24
hours
After
1000
documents
Client
Default
Associative Search Engine
Automatic pool update every
1
days
Starting at
12/31/2003
8:00:00 PM
Extended Settings
Enable Script debugging
Automatic Start
Min. free storage on disc:
30
MB
Min. free RAM:
30
MB
Enable batch integrity verification
OK
Cancel
Apply


Oracle Forms Recognition Runtime Service Properties
General
Workflow
Import
OCR+Export+Clean Up
Extended Processing
Input State:
Process Step:
Output State:
Import
succeeded
failed
100
50
100
OCR
succeeded
failed
200
150
200
Classification
classified
not classified
300
250
300
Extraction
valid
invalid
700
550
700
Export
valid
invalid
800
750
800
Clean Up
Perform advanced import failure processing
Perform folder based classification+extraction step
Perform folder based serial processing
Corrupted document's failure state:
0
OK
Cancel
Apply


Oracle Forms Recognition Runtime Service Properties
General
Workflow
Import
OCR+Export+Clean Up
Extended Processing
Import Directory:
C:\OFR Projects\AP\Import
...
Document Grouping
1 folder per batch (no subdirectories)
1 folder per document (no subdirectories)
1 batch per subdirectory, 1 folder per batch
1 batch per subdirectory, 1 folder per document
Document Type
Automatic
Skip:
Type:
tif
CI documents
Import Condition
Always import documents
Import only if ready file available in directory:
import.rdy
Import only if minimal number of documents are available in directory:
50
Import only if all files older than
Import if min. no. docs. available OR all files older than specified timespan
2
hours
Import only files which are older than specified timespan
Extended Settings
Limit batch size, max.:
10
documents
Import priority:
5
Batch prefix:
Batch_
Import *.wtx files
Folder prefix:
Folder_
File transfer mode:
Move files
Copy files
Additional File Import
Import 1 additional file per imported document:*
########.add
Import document only if additional file exists
Use alphanumeric characters + these wildcards to identify additional files:
* : will be replaced with the basename of the input file, e.g. "*.txt"
# : character at this position has to be identically in input and additional file
? : character at this position will be ignored
E-mail Import
Download e-mails from Exchange server using "RTS_Import" profile
Separate each part of MSG file into its own Workdoc
OK
Cancel
Apply



Once all four tabs have been populated, click the 'OK' button. The configuration of the runtime server instance is now complete.

6.8 Setting up the Verifier Application

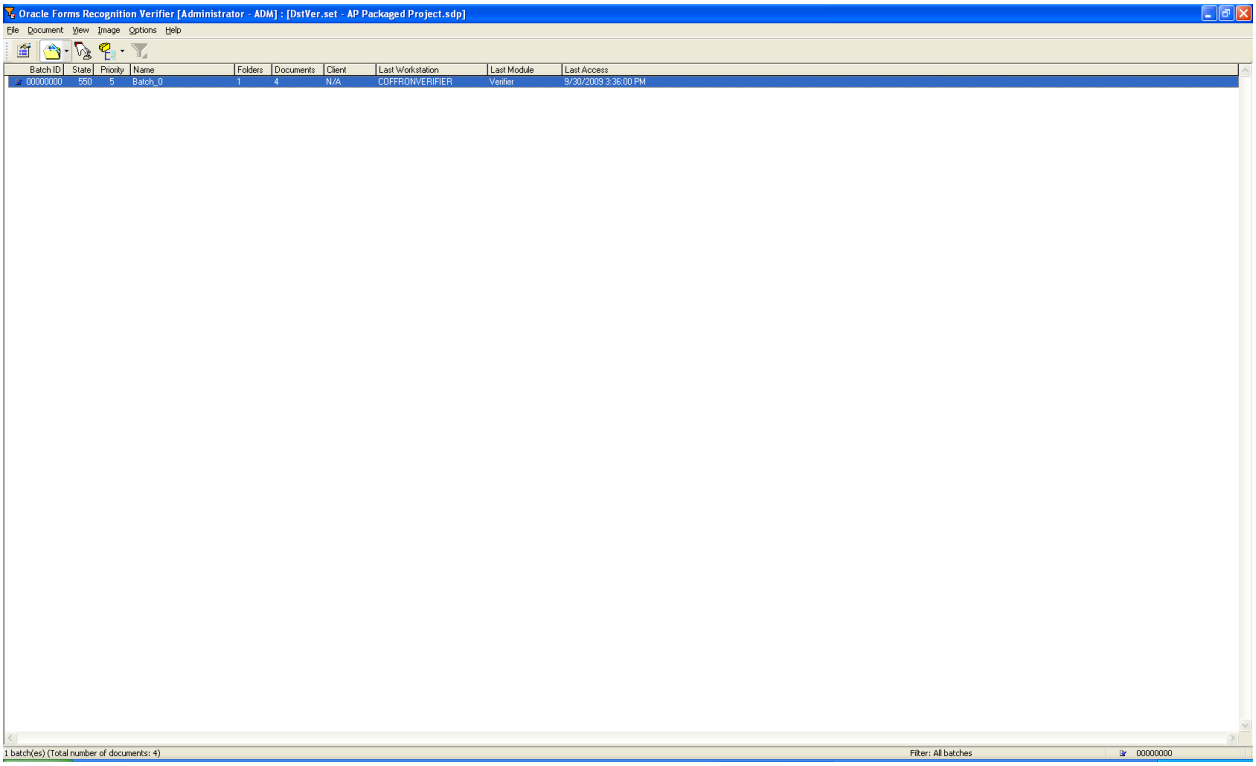
The Oracle Forms Recognition Verifier module is a thick-client application that can either sit directly on a user's local machine, or it can be installed on a central server and distributed to the users via Citrix.

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 (for example, the field was mandatory, but the system could not find it on the document; the field was extracted, but the system was not confident that the correct value had been ascertained), then the batch will be stopped in the Verifier application for a user to review.

To open the application if it is installed as a thick client on the local machine, select the following Windows menu path:

Start → Programs→ Oracle Forms Recognition→Oracle 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 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 (ready for export), but 1 has a status of 550 (extraction results require review), then the overall batch status will be 550.

A list of standard batch states, along with their corresponding meanings, can be found in table below:

Batch State	Meaning
0	Corrupt batch – unable to process
100	Documents have been imported successfully, and the batch is awaiting OCR
150	OCR step failed
200	Batch has been OCR'd, and is ready for classification
250	One or more documents in the batch failed to classify; manual intervention via Verifier is required
300	Batch is fully classified, and is ready for field extraction
550	One or more mandatory fields failed extraction/validation on one or more documents in the batch; manual intervention via Verifier is required
601-699	User designated exception states
700	All documents in the batch are valid from the point of view of classification and extraction; batch is ready for export
750	One or more documents in the batch failed the export step; manual intervention is required from the Oracle Forms Recognition system administrator
800	Batch successfully exported

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 system feels there is a problem with the field.

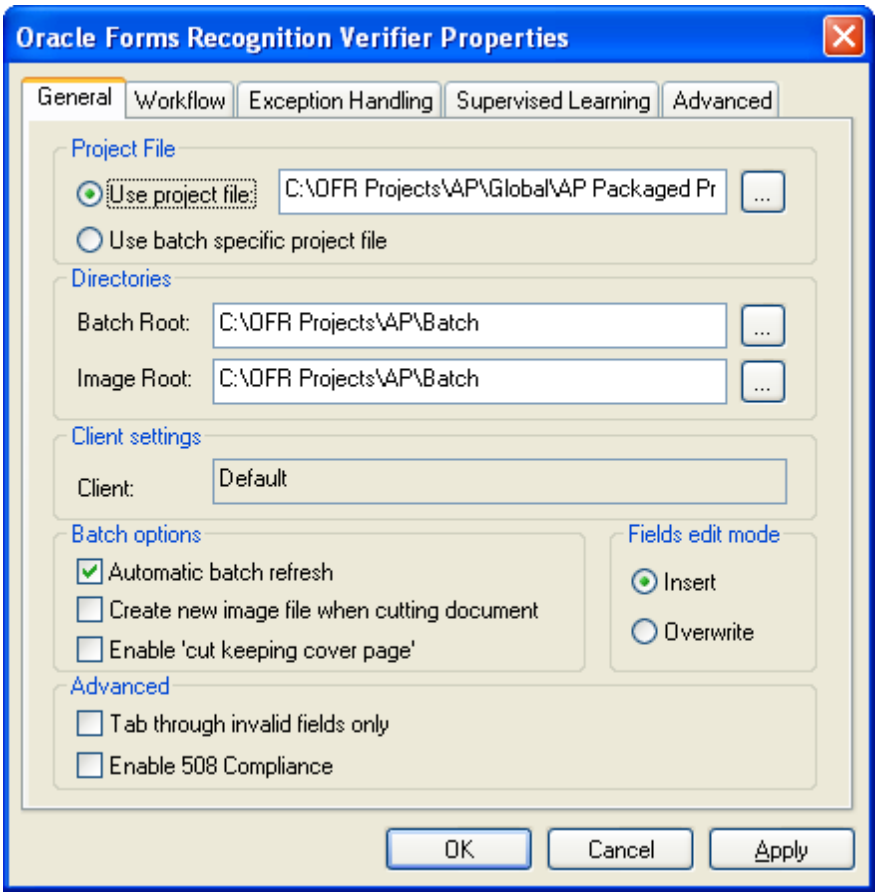
If the document is at a state of 750, which denotes a failed export, then the reason for the export failure is displayed as an error message against the invoice number field.

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 system configuration (see section 4.1.22).

To configure Verifier, open the settings dialog by selecting the menu path:

Options → Settings

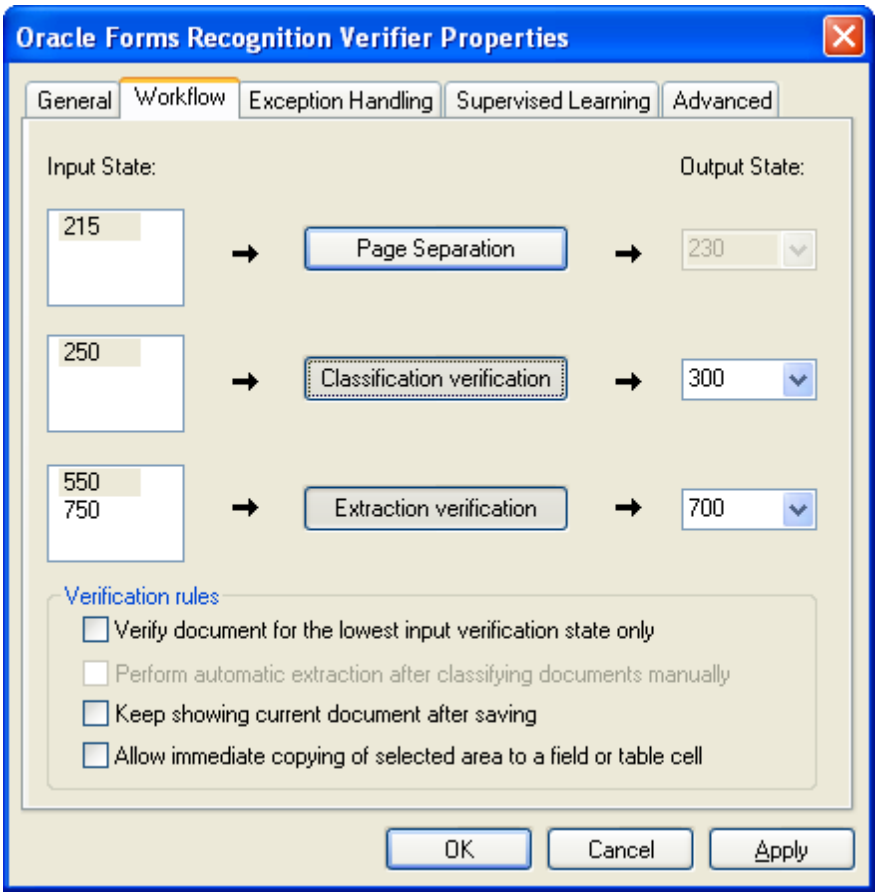
On the 'General' tab, enter the file paths to the Oracle Forms Recognition project file and the batch directory. An example screenshot is shown below:



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 set to ‘Overwrite’, 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.

On the ‘Workflow’ tab, relevant batch states for Verifier processing as specified. To add, remove or change an input state, simply right click on the input state window 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’, as per the screenshot below:



Once complete, click ‘OK’, and the system will return to the main batch list.

From here, select:

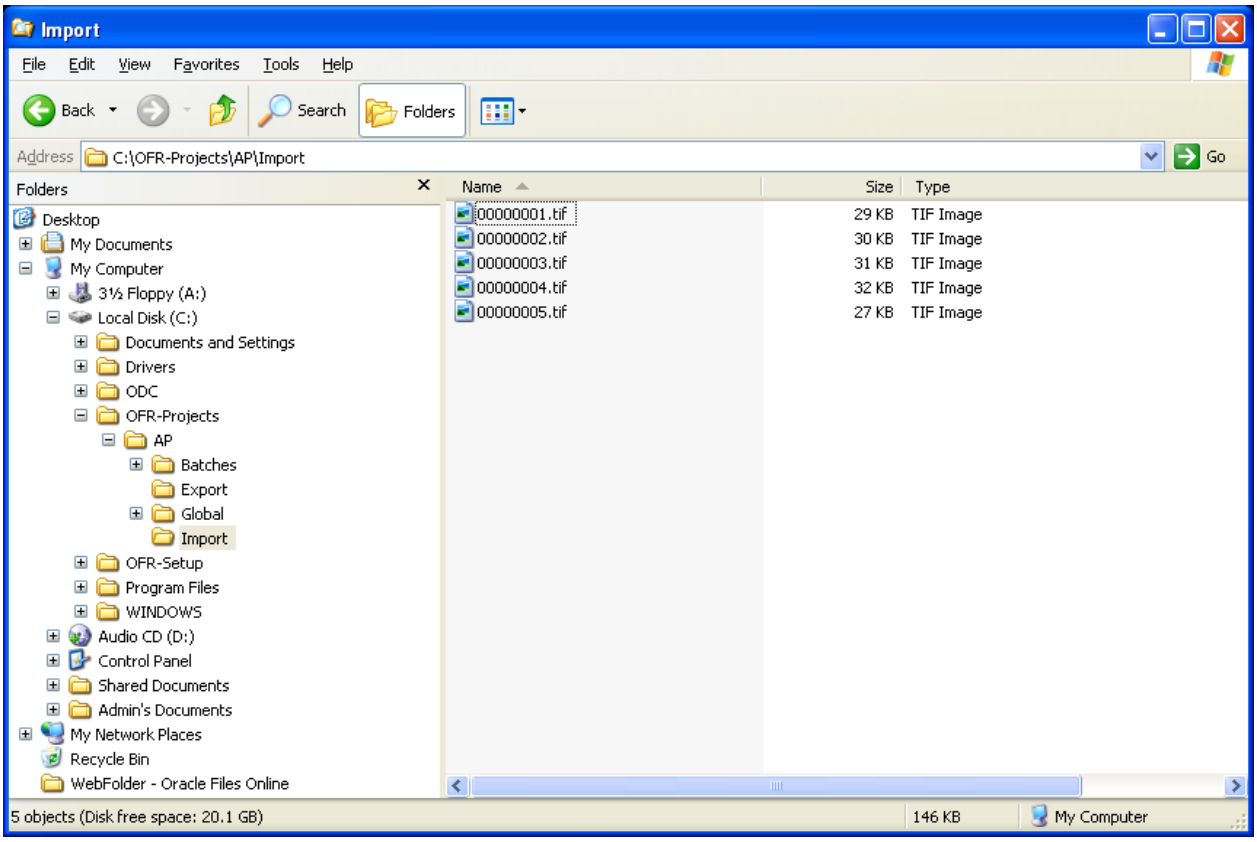
File → Save

The configuration of the Verifier application for a basic installation is now complete.

6.9 Processing Documents Through the System

At this point, the configuration of the system is complete, and it is now possible to process documents through the system.

To do this, place the appropriate documents into the Oracle Forms Recognition import directory created in section 6.2. In this example, four documents are going to be processed:

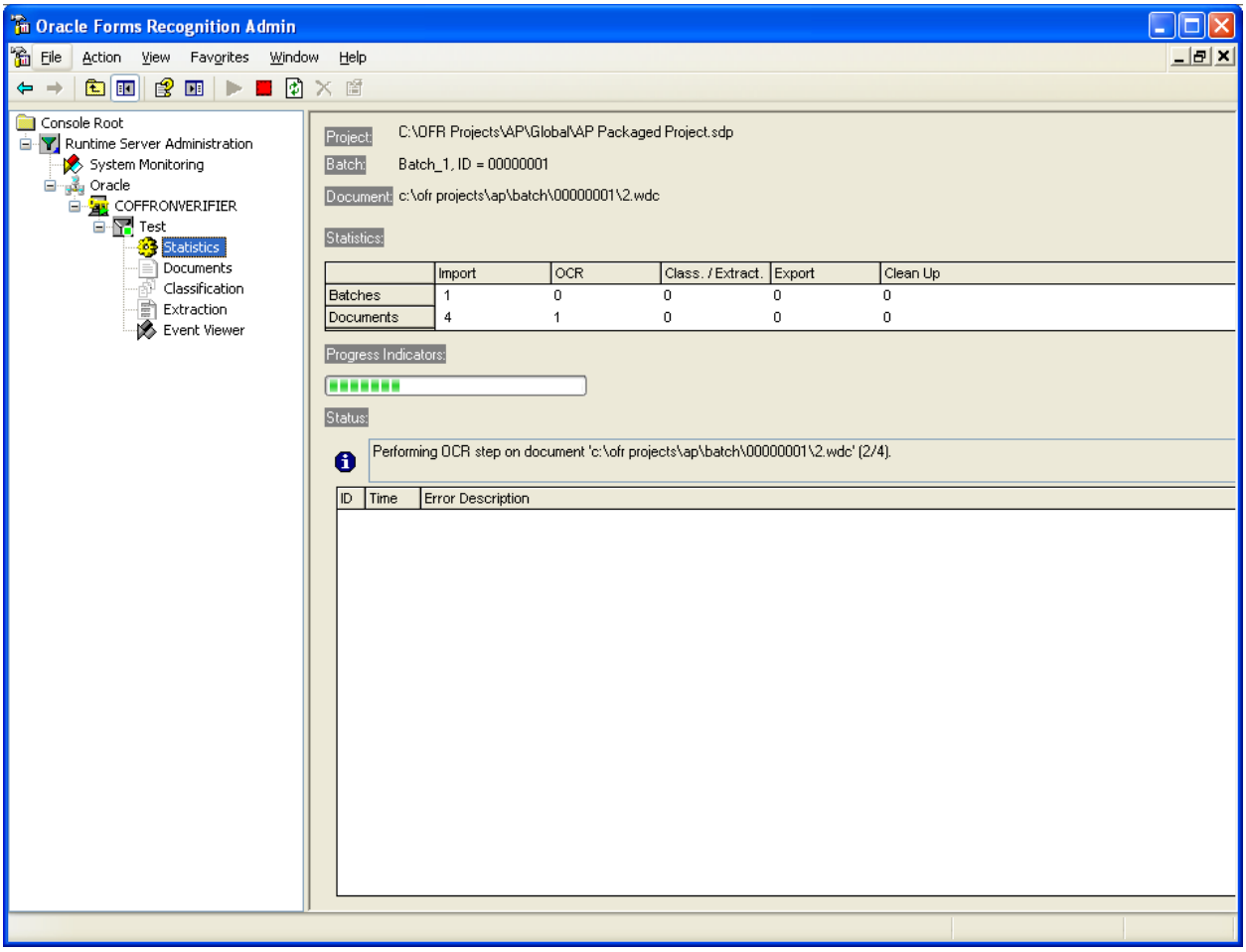


Oracle recommends that all image files processed by Oracle Forms Recognition be:

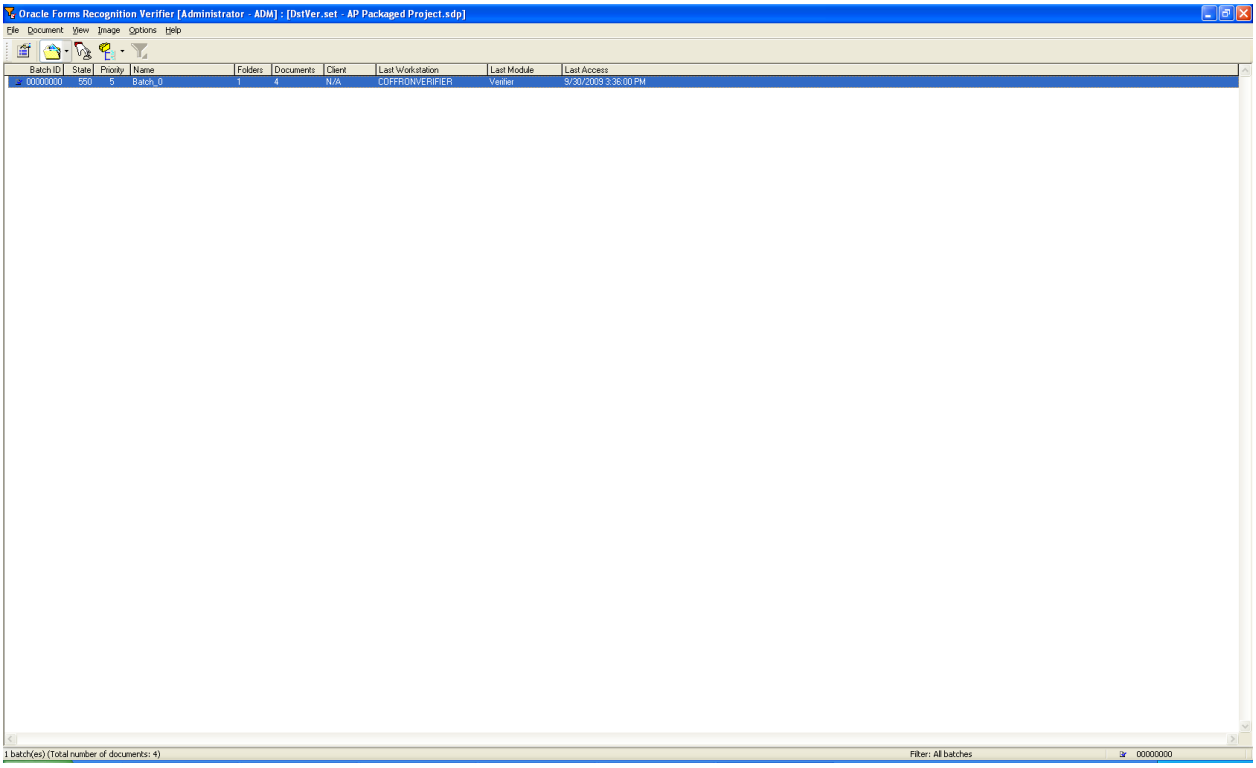
- TIFF with Group 4 compression
- 300 x 300 dpi
- Bitonal color depth (black and white)

If the document is a multi-page invoice, the file must be a multi-page TIFF with the pages in the correct order.

Now, open the Runtime Server management console, select the configured RTS instance, and click the green ‘play’ 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.



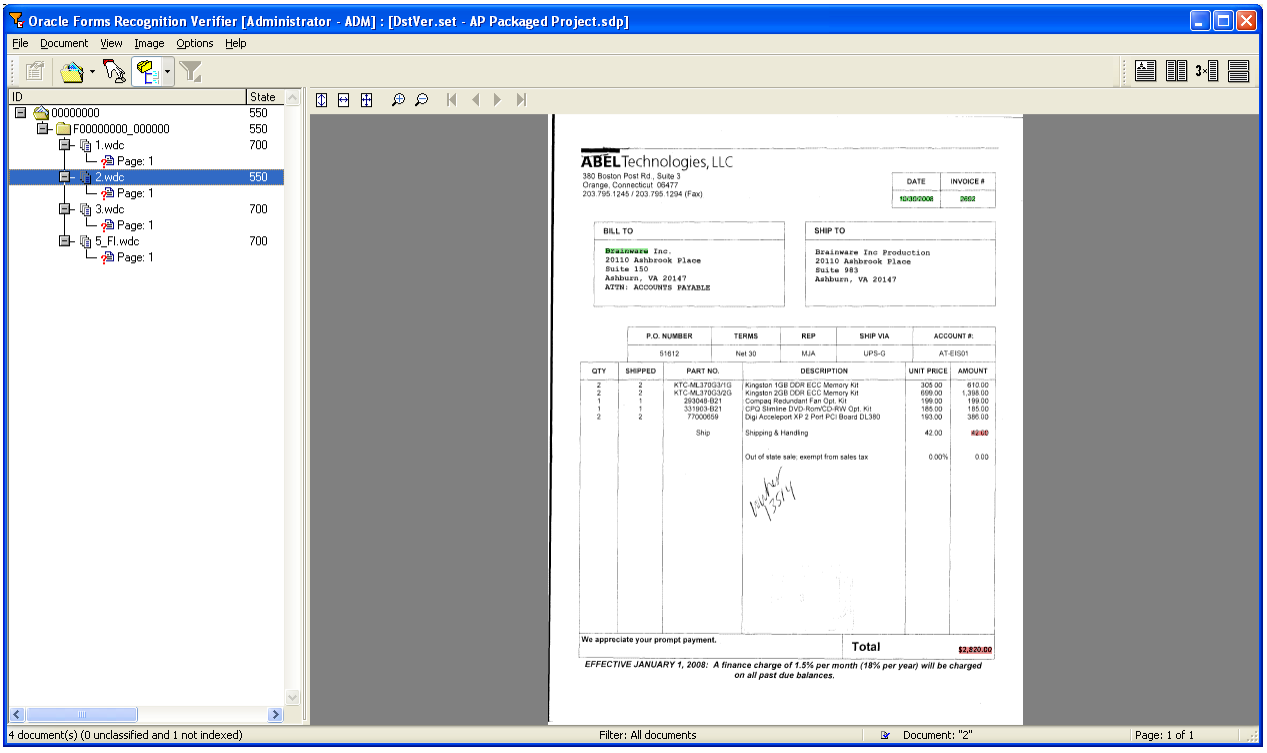
Once complete, if there was a problem with one or more documents, the batch will now appear in the Verifier application:



To get an overview as to how the system has performed, highlight the batch, and select:

View → Show Selected Batch

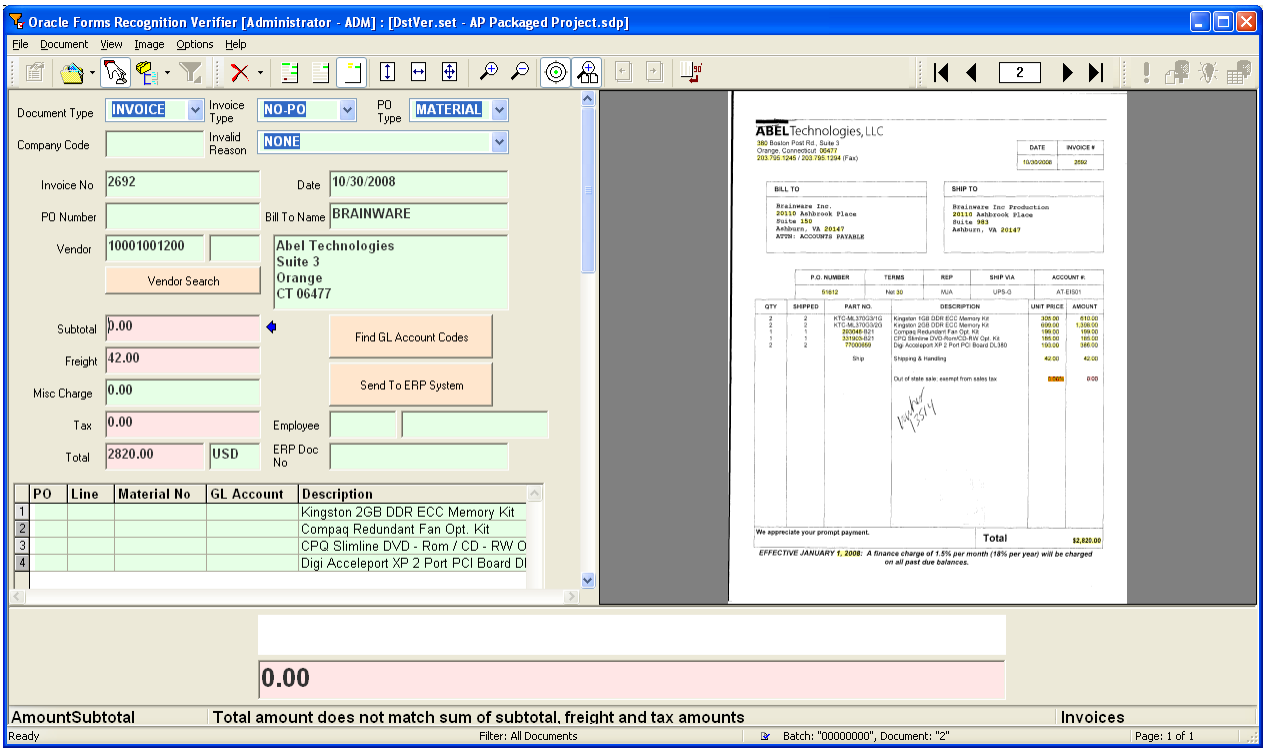
The batch overview screen will now be displayed:



This status screen provides an overview of all of the individual document states in the batch. In the example above, it can be seen that 3 of the 4 documents processed have a state of 700, meaning that the system was able to read and validate all of the required data. The second document in the batch has a status of 550, which means that there is a problem with the document for a user to investigate.

To enter the batch from this screen, select:

View → Verify Selected Batch.

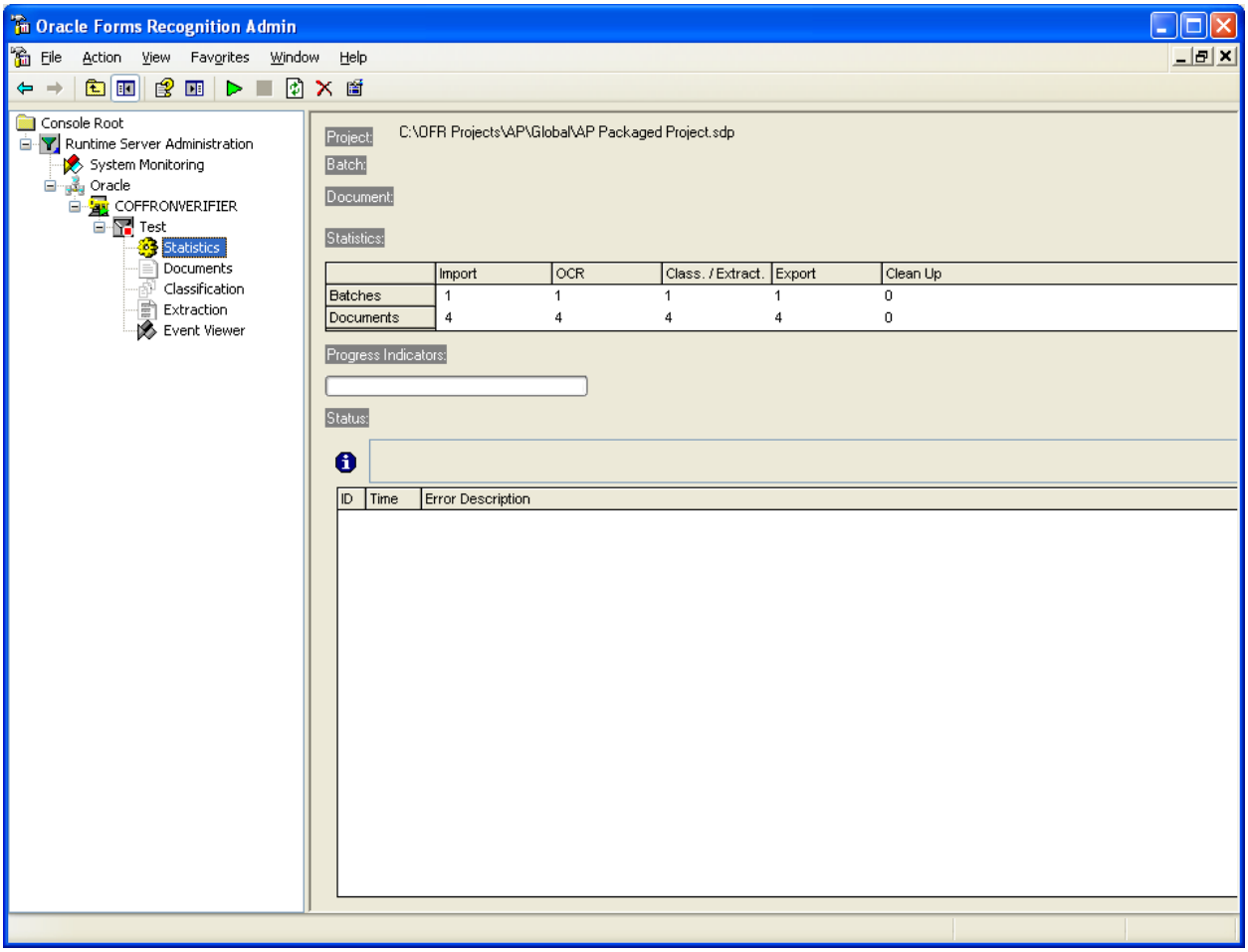


In the example above, the document has stopped with the invoice amount fields marked as invalid. The error message in the lower status bar informs the user that the total of the invoice line items extracted do not match up with the totals extracted at header level. In this scenario, the user should enter the missing line item.

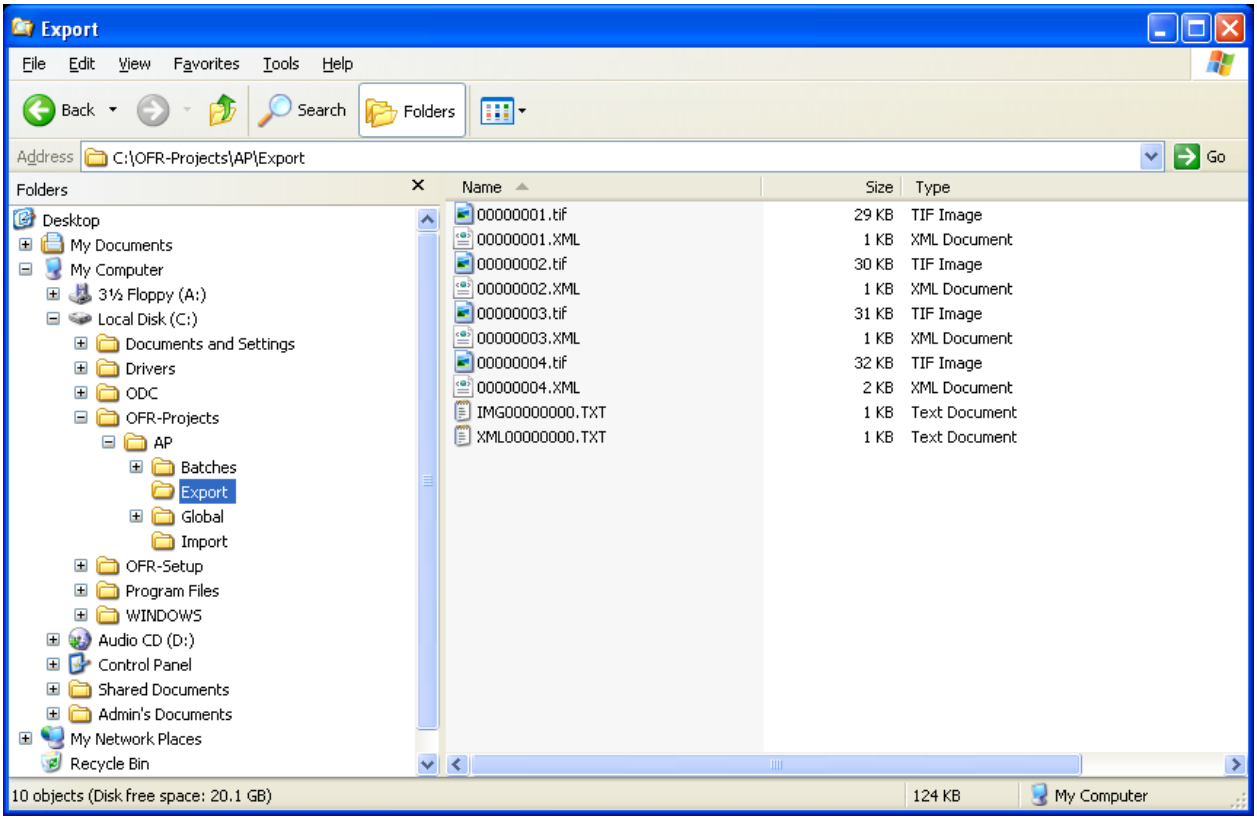
Once the problem has been corrected, the system will move onto the next invalid document in the batch, and so on until all documents in the batch are complete. At this point, the system will prompt the user to release the batch:

If the user selects 'YES', the batch state will be set to 700, which means that it is ready for export. Clicking the 'Details' button provides a radio button group which controls whether the system will move on to the next invalid batch, or whether the system will return to the batch overview screen.

Once ready for export, the Runtime Server instance will pick the batch back up again, and will carry out the export operation:



Once complete, navigate to the export directory configured in section 6.2. The directory will now contain a number of standard output files, one for each of the processed documents:



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 Oracle Forms Recognition A/P Solution.

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 instance settings contain an option for outputting the tiff image to the export directory, and this setting should be used over custom settings in the system 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 hold true:

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.

In the EXP section of the system configuration (see section 4.1.25), to output an additional tiff image, the following parameter 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 system configuration (see section 4.1.24), then the following parameter should be set to 'URN'.

```
EXP_VL_TiffName=URN
```

If left blank or set to anything else, 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 this parameter, the system will default to 300dpi.

To change the image compression type, the following options may be set against the ‘TiffFormat’ parameter:

Code	Compression type description
G4FAX	Standard Group 4 compression
G3FAX	Group 3 compression
LZWFAX	LZW compression
HUFFAX	HUF compression

For example, to select group 3 compression, the parameter should be set as follows:

```
EXP_VL_TiffFormat=G3FAX
```

If a blank or invalid entry is set against this 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 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=
```

If this 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 system configuration (see section 4.1.24), then 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 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=
```

If this 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 settings up a database export, and also 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 system configuration (see section 4.1.3) must be specified against the following parameter:

```
EXP_VL_SQLConnectionGroup=01
```

The database export can write into three different table types:

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 system configuration (see section 4.1.24) 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 now 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 'MYHEADERTABLE'.

Invoice header information can be written into the table as a fresh ‘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 either to ‘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 system configuration, there are over 40 header fields available to be mapped. Each of these has a corresponding parameter with an ‘EXP_VL_DBHC’ prefix. 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 – i.e.:

```
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: one is 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; the second is 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 (configured in the previous section)
```

In the above example, the system will write the line item data into database table ‘MYLINEITEMS’ where the document identifier is the ‘MYKEY’ column and the line item number field is ‘LINEID’. Hence, 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 out; 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 hold for any given document:

1) Line items are not subject to validation as set in the TAB section of the system configuration (see section 4.1.17).
2) The document is a credit memo, and line items are not required for credit notes.
3) The document is NO-PO, and line items are not required for NO-PO documents.
4) The document is PO-related, but the PO has not been released; therefore line items are not required for documents with an unreleased PO.
5) The user selected an 'INVOICE AMOUNTS DO NOT ADD UP' error code in Verifier.
6) The user selected either a 'VENDOR NOT FOUND', 'MISSING/INVALID PO' or a 'MISSING/INVALID VENDOR & 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 to the database 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 system configuration, there are over 25 line item fields available to be mapped. Each of these has a corresponding parameter with an 'EXP_VL_DBTable' prefix. 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 item 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 – i.e.:

```
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 clear out 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 also optional. The system expects the destination GL table to have a composite key consisting of two fields: one is 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; the second is 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 (configured in the previous section)
```

In the above example, the system will write the GL item data into database table 'MYGLACCOUNTS, where the document identifier is the 'MYKEY' column and the line item number field is 'GLLINEID'. Hence, 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 system configuration, there are over 25 GL item fields available to be mapped. Each of these has a corresponding parameter with an 'EXP_VL_DBGLTable' prefix. 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 – i.e.:

```
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 clear out 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.

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 database header export.

The process consists of two steps:

7.4.2.1 Adding a New .Ini File Parameter

The first step is to add a new parameter into the .ini file. This parameter, in common with the delivered database header export .ini file 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 .ini file.

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 .ini file thus:

```
EXP_VL_DBHCInvCode=INVOICE_CODE
```

7.4.2.2 Adding Script into the Database Export User Exit

The second and final step is to add a single line of script into the database header user exit.

To do this, open the project file with the Designer module; go to 'Definition mode'; and highlight the 'UserExits' class. Right click on the class and select 'Show Script' from the context menu.

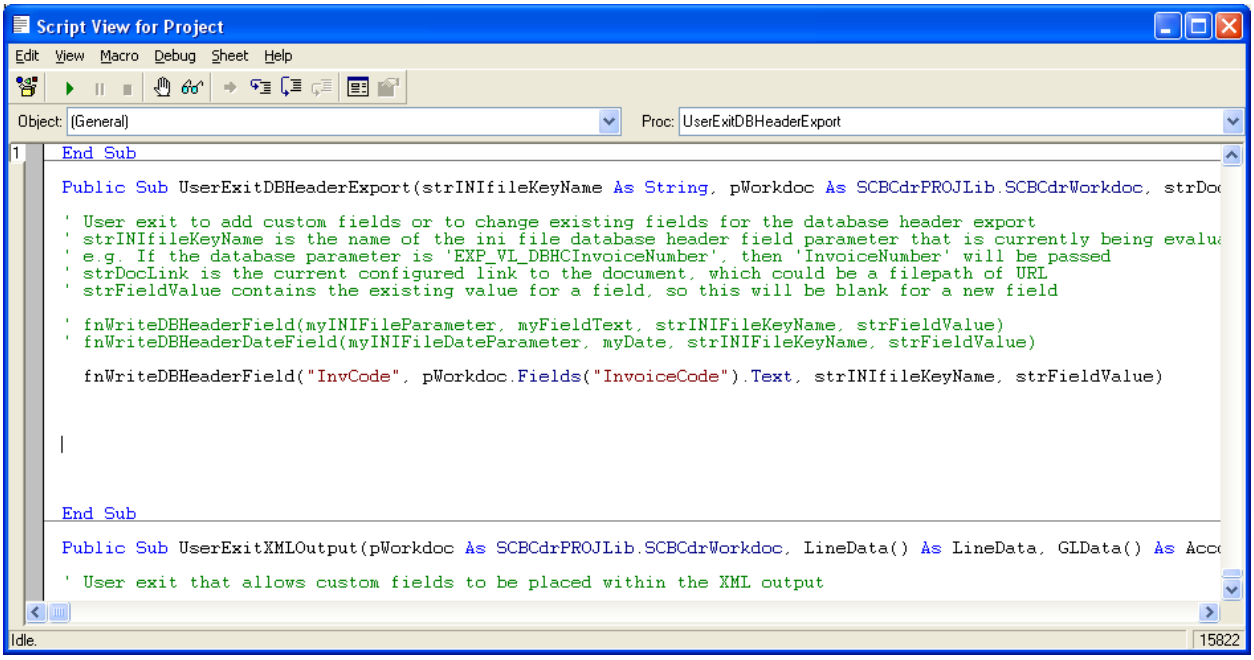
Once the script is open, navigate to the user exit sub-routine named 'UserExitDBHeaderExport'.

Now, add the following line into the sub-routine:

```
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. More information concerning this function can be found in section 5.2.1.2.

This is shown in the screenshot below:



Close the script and save the project.

The new field has now been added to the database export.

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 system configuration (see section 4.1.12), then 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 system configuration (see section 4.1.25).

7.5 Writing Data to an XML File

The following sections describe the process of configuring XML export, and also 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, invoice date, vendor ID, etc…

In the invoice line items section, the invoice line item information is written in 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 system 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:

```
<OracleDocument>    ← configured via EXP_VL_XMLFileHeader
      [Document level fields]
      <InvoiceHeader>   ← configured via EXP_VL_XMLInvoiceHeader
            [Invoice header fields]
      </InvoiceHeader>
      <InvoiceLines>    ← configured via EXP_VL_XMLLineItemsHeader
            [Invoice line item fields – one set of entries per line item]
      </InvoiceLines>
      <GLLines>   ← configured via EXP_VL_XMLGLLinesHeader
            [GL item fields – one set of entries per GL line]
      </GLLines>
</Oracle Document>
```

To activate output of an XML file, the following parameter must be set to 'YES' in the system 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 system configuration (see section 4.1.25), then 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 will default to '.XML'.

The XML file will always be written to the directory specified as the export directory on 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=C:\OFR Projects\AP\Export
```

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 also define how they are tagged.

In the system 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 hold for any given document:

1) Line items are not subject to validation as set in the TAB section of the system configuration (see section 4.1.18).
2) The document is a credit memo, and line items are not required for credit notes.
3) The document is NO-PO, and line items are not required for NO-PO documents.
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' error code in Verifier.

6) The user selected either a ‘VENDOR NOT FOUND’, ‘MISSING/INVALID PO’ or a ‘MISSING/INVALID VENDOR & 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 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 two steps:

7.5.2.1 Adding a New .Ini File Parameter

The first step is to add a new parameter into the .ini file. This parameter, in common with the delivered XML invoice header .ini file 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 .ini file.

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 .ini file thus:

```
EXP_VL_XMLHCInvoiceCode=InvCode
```

7.5.2.2 Adding Script into the XML Export User Exit

The second and final step is to add a single line of script into the XML output user exit.

To do this, open the project file with the Designer module, then go to ‘Definition mode’, and highlight the ‘UserExits’ class. Right click on the class and select ‘Show Script’ from the context menu.

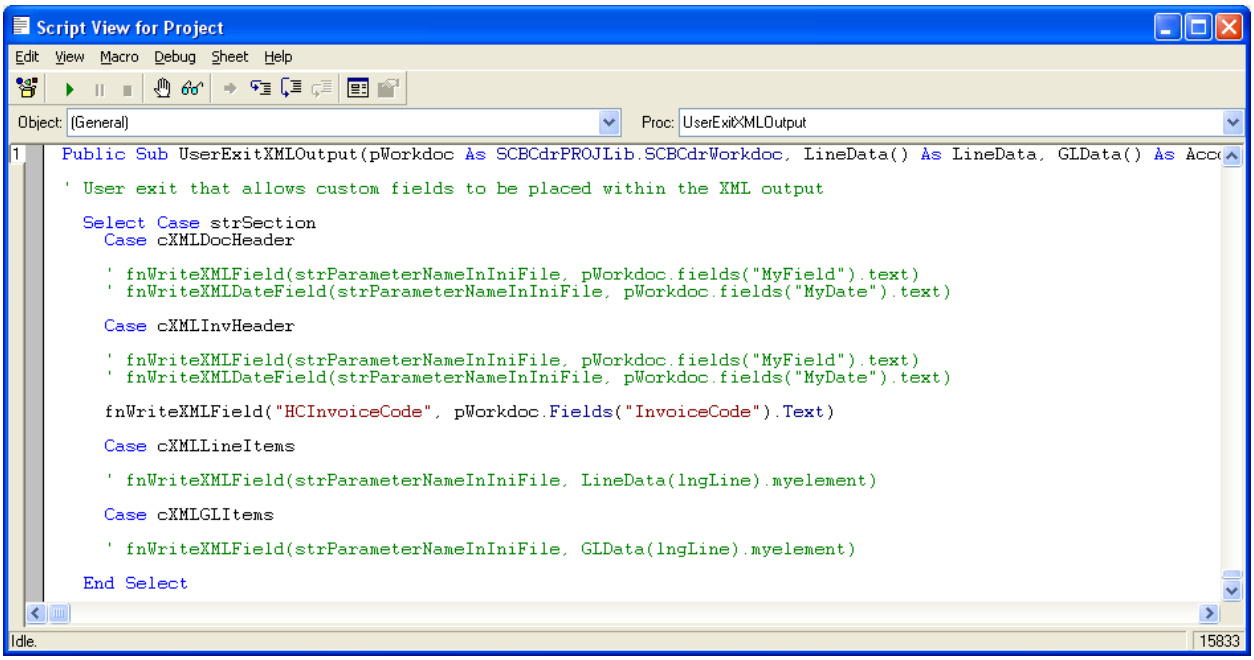
Once the script is open, navigate to the user exit sub-routine named ‘UserExitXMLOutput’.

Now, add the following line into the sub-routine. This line should go in the invoice header section of the sub-routine’s select case structure.

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

The ‘fnWriteXMLField’ function has been specially provided for this purpose to condense the operation into a single command. More information concerning this function can be found in section 5.2.1.2.

This is shown in the screenshot below:



Close the script 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’, then it will appear as follows:

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

When specifying the .ini file parameter, only the ‘XML’ prefix can be dropped.

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 pass 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 system configuration (see section 4.1.11), 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 system configuration (see section 4.1.26).

7.5.2.3 Inserting Additional Fields Derived From the Document Filename

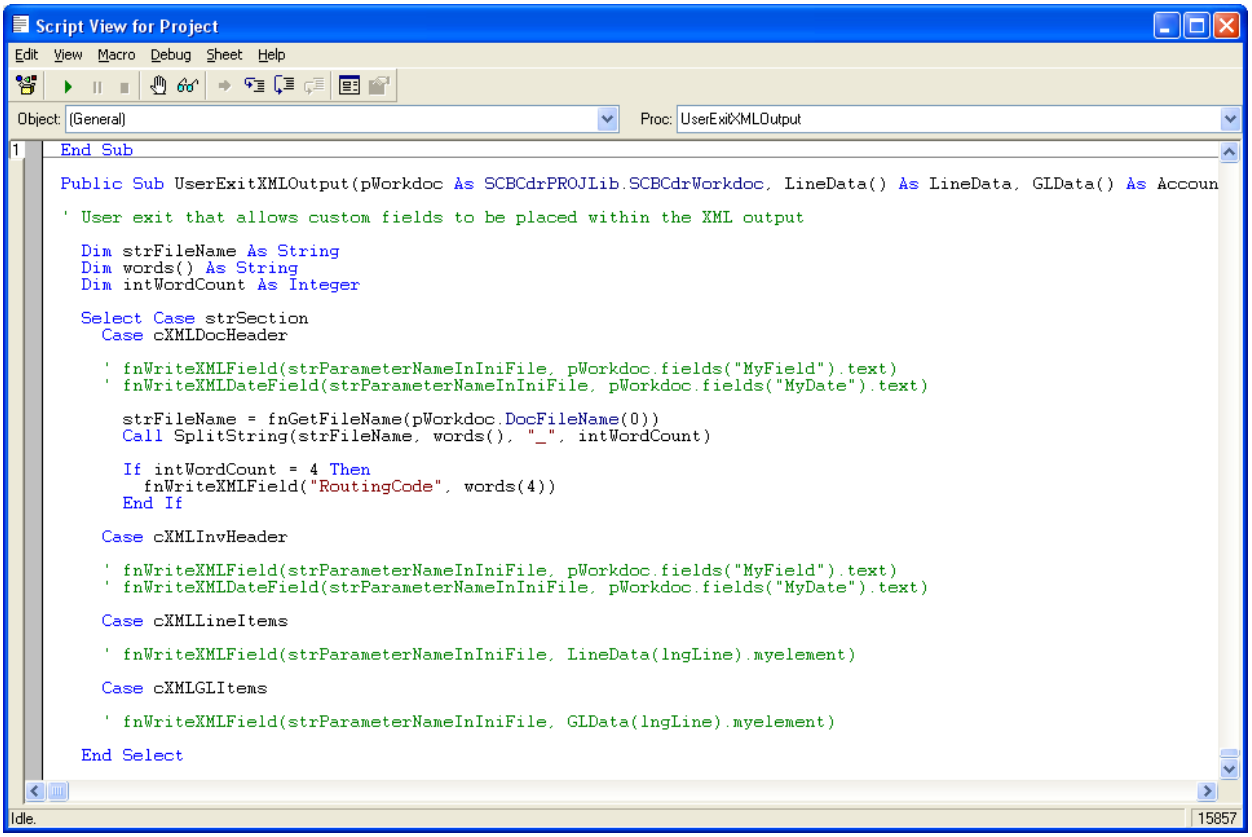
Additional values inserted into the XML file need not be tied to actual fields within the project file. It is also possible to insert hard-coded values, or to derive values from properties of the document – for example, from the document filename.

For example, the document filename is 12345_ABCD_20090901_ACD.tif. The business requirement is that the 'ACD' component of the filename, which denotes a special accounting routing code, is passed into the XML file in the document header section.

In this instance, there is no corresponding parameter in the IMP section of the system configuration which denotes a special accounting routing code, hence a custom solution is required.

Custom parameter 'EXP_VL_XMLRoutingCode=RoutingCode' has already been added into the .ini file.

The script example in the screenshot below shows how the global subroutine 'SplitString' and the global function 'fnGetFileName' can be used to pass an additional parameter from the filename into the XML document header section. These functions and their usage are described in greater detail in section 5.2.1.2.



When the document is processed, the following line will be written into the document header section of the XML file:

```
<RoutingCode>ACD</RoutingCode>
```

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 the 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 system configuration (see section 4.1.30).

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 'one-stop' switch to activate/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 system configuration will achieve this.

To activate CSV file output for group '01', the following parameter, which is the 'local' switch for the group, 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 Oracle Forms Recognition workflow. If no export directory is specified there either, the default export file path set in the EXP section of the system 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, which is controlled via the 'CombinedFilePerBatch' parameter which should be set to 'YES' if one file per batch is required; if one CSV file per document is required, it should be set to 'NO'.

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 take the following naming convention:

PPPPYYYY.ZZZ

In the above, 'PPPP' is the file prefix set against to the group 'FilePrefix' parameter; 'YYYY' is either the original name of the image file or the document URN mapped in the IMP section of the system configuration, which is controlled by setting the group 'Filename' parameter; and, 'ZZZ' is the file extension set against the group 'FileType' parameter.

For a CSV file per batch, the filename will take the following naming convention:

PPPPBBBBBBBB.ZZZ

In the above, ‘PPPP’ is the file prefix set against the group ‘FilePrefix’ parameter; ‘BBBBBBBB’ is the numeric 8-digit batch ID; and, ‘ZZZ’ is the file extension set against the group ‘FileType’ parameter.

Example 1:

CSV group ’01’ is configured as follows:

```
CSV_OP_01_OutputFile=YES
CSV_OP_01_CombinedFilePerBatch=NO
CSV_OP_01_Filepath=
CSV_VL_01_Filename=
CSV_VL_01_FileType=
CSV_VL_01_FilePrefix=XML_
```

Using this configuration, if the document being processed was ‘1234.tif’, the output file would be named: XML_1234.CSV.

Example 2:

CSV group ‘02’ is configured as follows:

```
CSV_OP_02_OutputFile=YES
CSV_OP_02_CombinedFilePerBatch=NO
CSV_OP_02_Filepath=\\mydirectory
CSV_VL_02_Filename=URN
CSV_VL_02_FileType=LOG
CSV_VL_02_FilePrefix=BW_
```

Using this configuration, if the document being processed was ‘1234_ABC_5678.tif’ and COMPONENT2 was mapped as the document URN in the IMP section of the system configuration, the output file would be named: \\mydirectory\BW_ABC.log.

Example 3:

CSV group ‘03’ is configured as follows:

```
CSV_OP_03_OutputFile=YES
CSV_OP_03_CombinedFilePerBatch=YES
CSV_OP_03_Filepath=
CSV_VL_03_Filename=
CSV_VL_03_FileType=
CSV_VL_03_FilePrefix=BW_
```

Using this configuration, for processing of batch ‘00000010’, the output file would be named: BW_00000010.CSV.

It is also possible to specify a date format and separator for each individual CSV file outputted. Date formats can be either ‘YYYYMMDD’, ‘MMDDYYYY’ or ‘DDMMYYYY’. If no date format is

specified then the output date format configured in the EXP section of the system 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 was 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 standard '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 set to 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 'FormatLine*N*' 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 denotes the end of the header record entry.

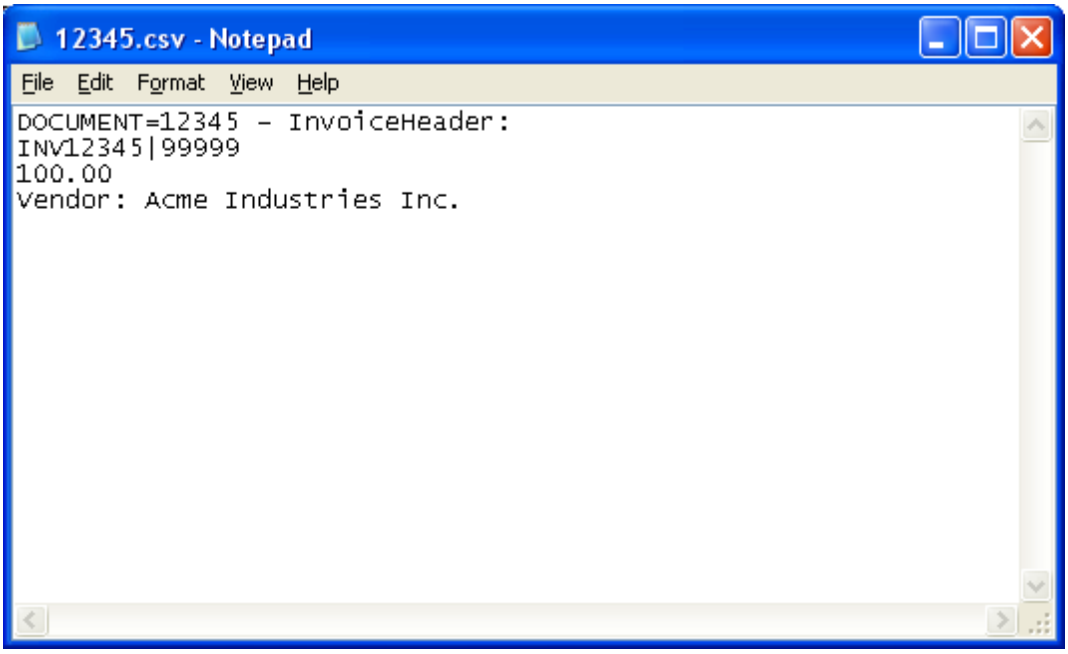
Values set against the 'FormatLine' parameters are essentially freetext, where literals (for example, <%INO> representing the invoice number) are used to specify where extracted values from the document should be inserted into that particular line. A full list of available literals can be found in section 4.1.30.

Example:

The line format parameters are set as follows:

```
CSV_VL_01_FormatLine1=DOCUMENT=<%TNF> - InvoiceHeader:
CSV_VL_01_FormatLine2=<%INO>|<%PON>
CSV_VL_01_FormatLine3=<%TOT>
CSV_VL_01_FormatLine4=Vendor: <%VNM>
CSV_VL_01_FormatLine5=
```

The incoming document 12345.tif has an invoice number of 'INV12345', a purchase order number of '99999', a total amount of '100.00' and the vendor is Acme Industries Inc. The CSV file would be generated as per the screenshot below:



If a more standard CSV file is required that uses a simple separator, 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. A full list of available literals can be found in section 4.1.30.

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, open the project file with the Designer module, then go to 'Definition mode', and highlight the 'UserExits' class. Right click on the class and select 'Show Script' from the context menu.

Once the script is open, navigate to the user exit sub-routine named 'UserExitCSVFile'.

To add custom field 'InvoiceCode' into the file, add the following line into the sub-routine:

```
fnWriteCSVField(strRecordText, strKey, "<%ZIC>",
pWorkdoc.Fields("InvoiceCode").Text)
```

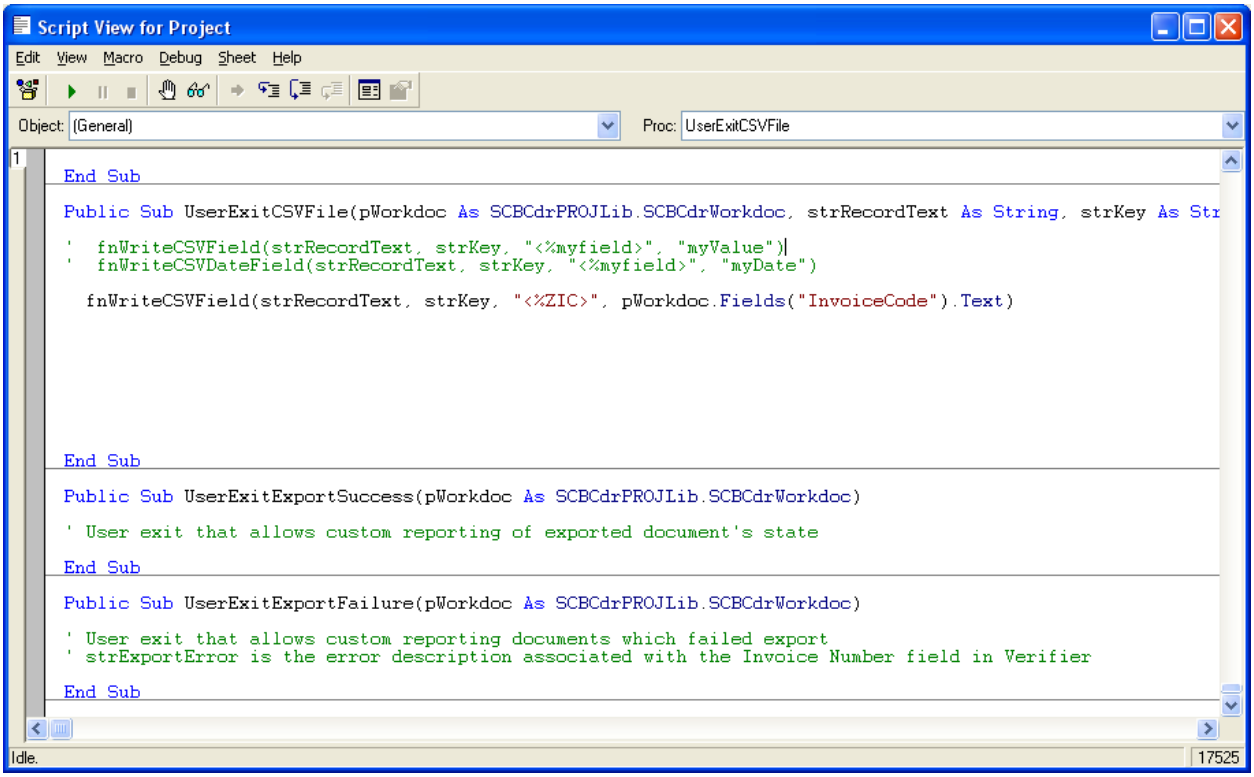
The 'fnWriteCSVField function has been provided to condense this operation into a single command. More information concerning this function can be found in section 5.2.1.2.

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 additional standard system

literals added in future versions of the A/P packaged project. '<%ZIC>' should subsequently be included in the CSV group line format parameters so that it is written into the file.

Example: `CSV_VL_01_FormatLine1=<%ZIC>`

The sample code in the user exit is shown in the screenshot below:



Close the script and save the project.

The new field has now been added to the CSV file output.

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 system configuration (see section 4.1.11), 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 system configuration (see section 4.1.25).

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, open the project file with the Designer module, then go to 'Definition mode', and highlight the 'UserExits' class. Right click on the class and select 'Show Script' from the context menu.

Once the script is open, navigate to the user exit sub-routine named 'UserExitCSVFileLine'.

This user exit works the same way as ‘UserExitCSVFile’, except that a structure (‘LineData’) containing the current line item to be written into the file is included in the interface.

The components of the LineData structure are described in section 5.2.5.1.

Example usage:

A client requires the subsequent credit/debit indicator to be passed into the file at line item level. This value is available within the LineData structure, but only as ‘X’ or blank. The client requires this value to be written into the file as ‘SUB’ or ‘NOSUB’ respectively.

The first step is to define a literal to represent the customized value. In this case, ‘<%ZSD>’ has been decided upon. It is recommended to use a ‘Z’ as the first character of the literal so that there will be no conflicts with any additional standard system literals added in future versions of the solution. ‘<%ZSD>’ should subsequently be included in the CSV group line item format parameter so that it is written into the file, in this case, after the PO number, the PO line item number, the quantity and the line item amount, separated by a hyphen.

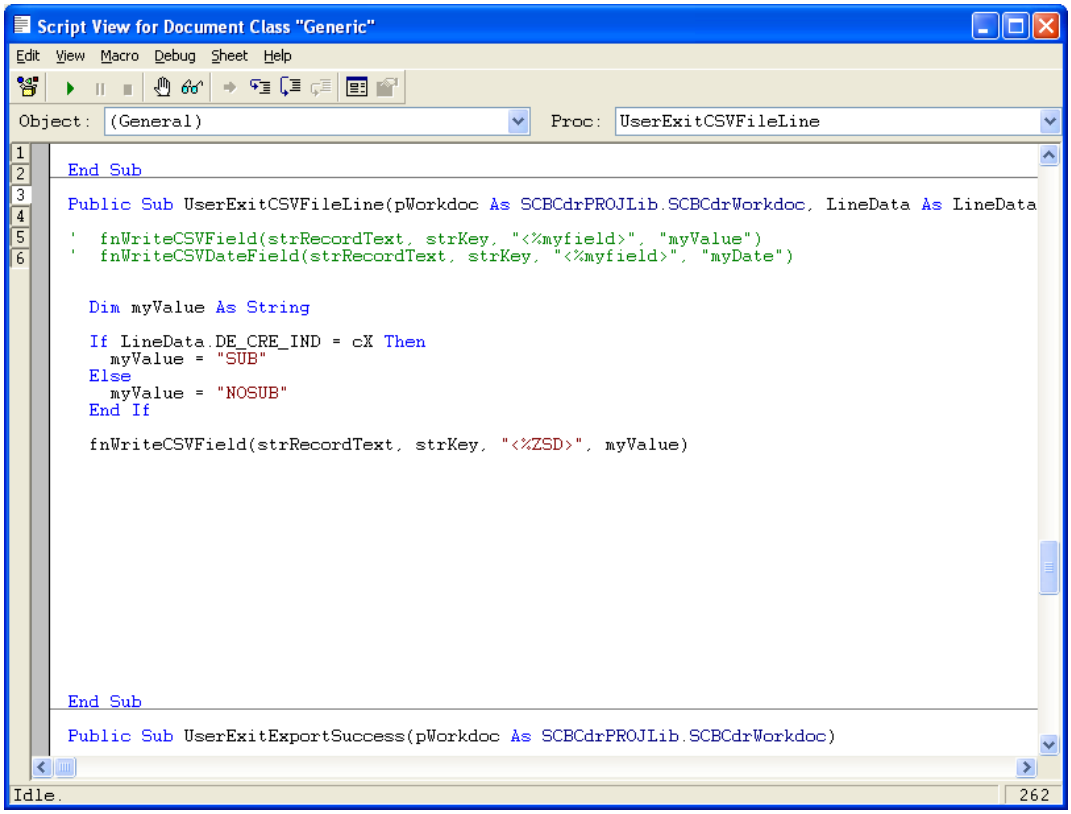
Example: `CSV_VL_01_LineItem=<%LPO>-<%LPL>-<%LQT>-<%LTO>-<%ZSD>`

As a hyphen is to be used as the separator, this should be registered against the CSV file group separator parameter:

```
CSV_VL_01_Separator=-
```

This ensures that none of the values to be written into the file contain a hyphen, which would otherwise create an uneven number of columns.

The final step would be to add the following code into ‘UserExitCSVFileLine’, as per the screenshot below:



Close the script and save the project.

The custom value has now been added to the CSV file line item output.

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.

To gain access to the user exit, open the project file with the Designer module, then go to 'Definition mode', and highlight the 'UserExits' class. Right click on the class and select 'Show Script' from the context menu.

Once the script is open, navigate to the user exit sub-routine named 'UserExitCustomExport'.

This sub-routine receives the following parameters:

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 system configuration parameter 'EXP_VL_DefaultExportPath'.
strDocLink	Path to the image of the document, which could be stored either in a storage director 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. i.e. where The structure of the array is described in section 5.2.5.1.
GLData	Array based on the accounting data type defined on the global variables 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 extracted either at header or line item level that is configured to be posted to a GL account in the MSC section of the system configuration. The structure of the array is described in section 5.2.5.7.
TaxData	Array containing the total tax amount that corresponds to each tax code determined during the automatic tax calculation procedure. The structure of the array is described in section 5.2.5.3.
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. The definition of the address structure is described in section 5.2.5.4.
Flags	Document validation flags This structure contains document-specific flags which can be used to determine what data should be exported. The definition of the structure is described in section 5.2.5.6.

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 actually exist against the document class, then a runtime error will be thrown. Global function 'fnGetBaseClass' (described in section 5.2.1.2) is provided for this purpose.

To activate the custom export, the following parameter in the EXP section of the system configuration (see section 4.1.25) must be set to 'YES':

```
EXP_OP_CustomExport=YES
```

If an error occurs during export, then populating global variable 'strExportError' with a descriptive error message will have the effect of failing the export (i.e. sending 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. The history can be cleared by resetting the document back to state 200.

If the export is not successful, the user exit will be called again upon the next attempt.

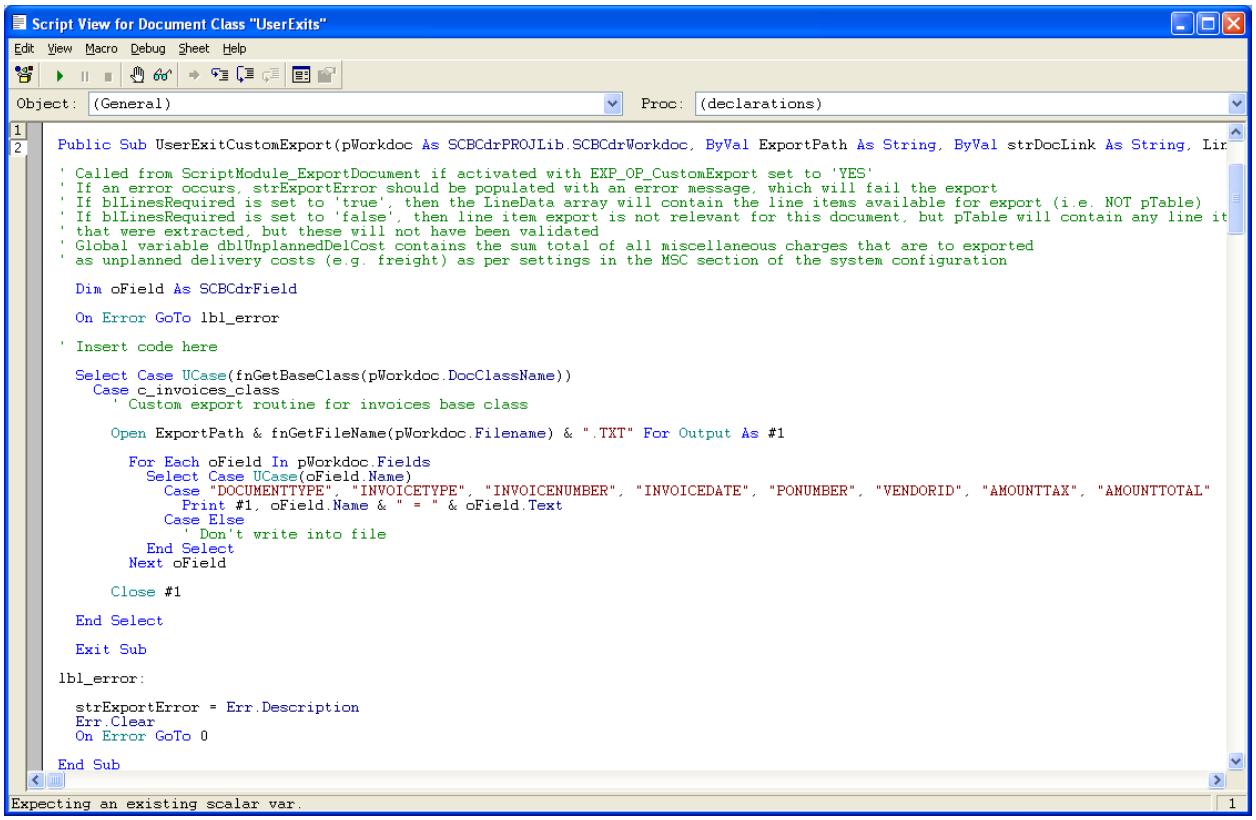
The history check can be over-ridden by setting the following parameter to 'YES' in the system configuration:

```
EXP_OP_RedoAllExports=YES
```

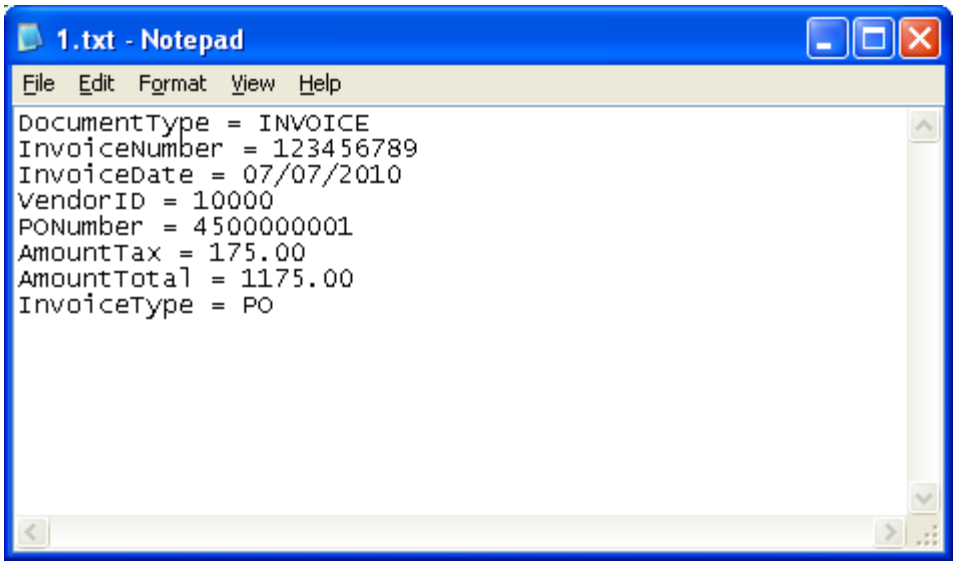
This is not recommended as a 'business as usual' setting in a production system.

7.7.2 Custom Export of Header Data

The following example shows how header data may be exported for documents classified to the 'INVOICES' base class. It writes header field information into a simple text file. A 'Select' statement has been used in this instance in order to restrict output to a set list of fields.



When the data is exported, if the filename was '1.tif', the following flat file will be created in the document export directory:



Special care needs to be taken with the following header fields:

Header field	Comments

AmountTotal	When exporting the total invoice value, care must be taken to add back on the invoice withholding tax amount.
	
	Hence, the correct calculation of the invoice total is:
	
	`dblTotal = fnConvertToDouble(pWorkdoc.Fields("AmountTotal").Text) + fnConvertToDouble(pWorkdoc.Fields("AmountWithholdingTax").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 should be observed:
	
	`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 (described in section 5.2.5.6) 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"`
	
	`Else`
	
	`  ' amounts confirmed in Verifier`
	
	`  strOutputTotal = Replace(CStr(dblTotal), ",", ".")`

	End If
AmountTax	Within the project file, three 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) and 'ICMS' (Brazilian ICMS tax). During export, all three should be summed together and passed as the total document tax, via the following code: `dblTax = fnConvertToDouble(pWorkdoc.Fields("AmountTax").Text) + fnConvertToDouble(pWorkdoc.Fields("PST").Text) + fnConvertToDouble(pWorkdoc.Fields("ICMS").Text)` No tax should be exported if automatic tax determinations is being used to

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 system 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 client has three types of miscellaneous charges: freight, pallet charges and customs charges. In the MSC section of the system 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 code:
	
	`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")`
	
	`End If`
	
	If the above code 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. Hence, the standard 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 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: Client 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: Client used 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: Client 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. A definition of the address structure can be found in section 5.2.5.4.
	
	For example, the following code 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 lifted from the standard 'PONumber' field.
	
	In Verifier, it is possible that a user selected a 'Missing/Invalid PO' or 'Missing/Invalid Vendor & PO' invalid reason, so the content of the field may not be correct.
	
	The 'NOPO' property of the 'Flags' object (described in section 5.2.5.6) provides an easy way to detect this. For example:
	
	`Dim strOutputPO As String`
	
	`If Flags.NoPO Then`
	
	`  ' 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. **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 pounds sterling as well as in US dollars, as pounds sterling is the local currency of the UK, where the VAT is being levied. 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.
	
	Hence, 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' field 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.
	
	Here is 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 1-based. The structure of the array is described in section 5.2.5.1.

No line items with a value of zero will be included within the array, except in a very special case where line pairing is switched on 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 switched on or off.

If line pairing is switched on, 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 the system was able to post against planned conditions;
4) All miscellaneous charges that were set to post as a specific line type.

Paired line 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; 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'.

The array will not contain any miscellaneous charges that were set to post against a planned condition, but the correct condition could not be identified. It will also not contain any lines for miscellaneous charges where no output type had been configured.

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'.

The processing behavior of miscellaneous charges (and how to configure it) is described further in Appendix E: Configuring the Miscellaneous Charges.

If line pairing is switched off, 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 system configuration control the types of document and circumstances that line item extraction is relevant for.

Within 'UserExitCustomExport', Boolean import parameter 'blLinesRequired' provides a simple 'go/no-go' indicator as to whether line items need to be exported for the current document.

The following code 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 the first 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
======
```

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

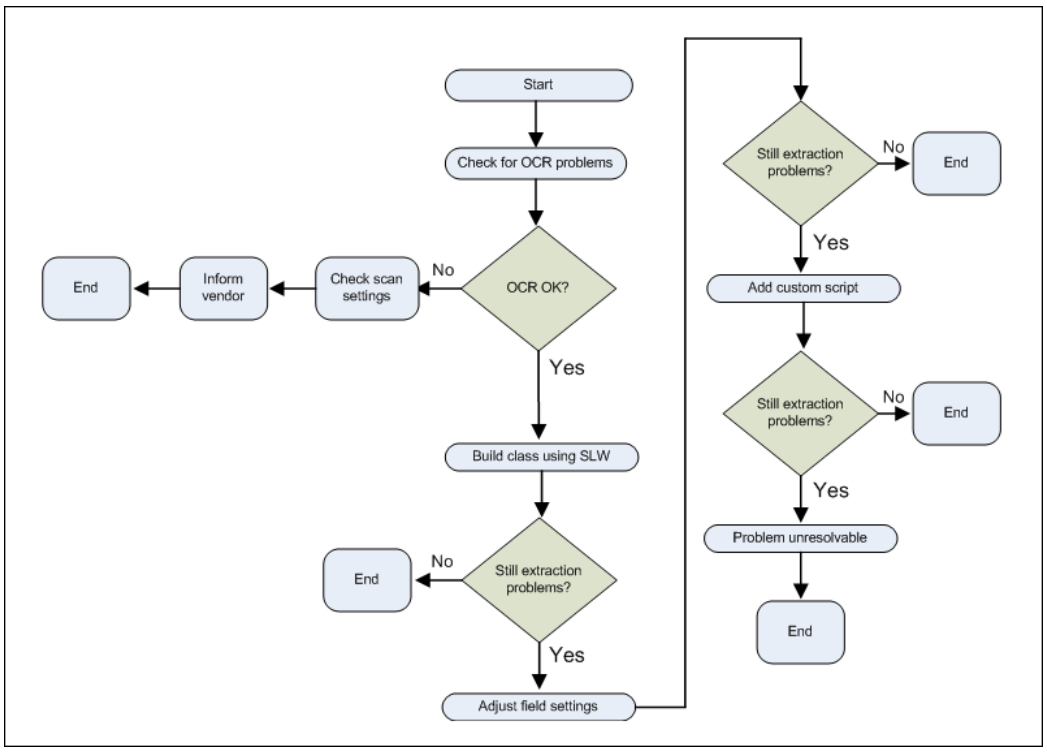
Oracle Forms Recognition comes pre-configured with a generic learnset for the extraction of data from invoices. This means that out-of-the-box, the system should already be able to deliver high, industry-leading extraction results from invoice documents of any format without any special configuration.

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 Oracle Forms Recognition 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.

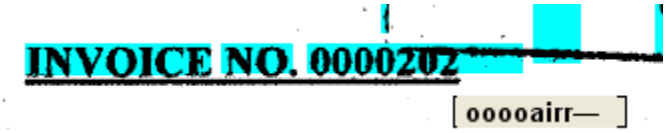
If the document OCR is particularly poor to the point where the information that required extracting was compromised, then the document is not a good candidate for further improvement.

The following sections describe the two different varieties of OCR error that may occur, and suggest possible courses of action that may be taken.

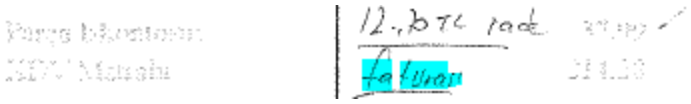
8.2.1.1 OCR Problems on the Field Value

OCR problems with the field value itself may be caused by:

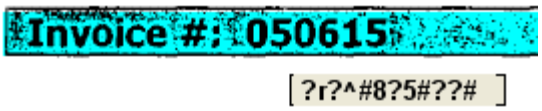
1) Handwriting or stamps obscuring the required data;



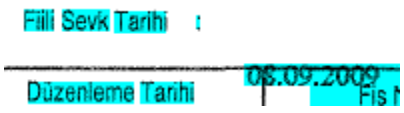
2) Poor invoice print quality (for example, a dot matrix print ribbon running out of ink);



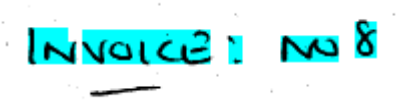
3) Shading on the invoice (for example, black lettering on a dark grey background);



4) Misalignment of print text against a template background;



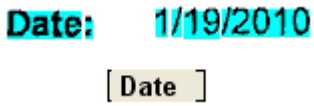
5) Handwritten information



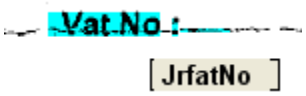
Checking the invoice OCR read can be done in both the Designer and Verifier applications by hovering the cursor over the area of the document where the correct field value is located.

In Designer, when viewing a document, clicking the 'Highlight All Words' button (shown below) on the top toolbar will show the OCR results for the entire document.

Each green block represents a single OCR word. Positioning the cursor over each green block shows how the system has interpreted the word:



In the above example, 'Date' has been read correctly. In the example below, 'Vat No' has been read incorrectly as 'JrfatNo' due to a line running across the page, which may have been caused by a roller on the scanner that requires cleaning:



There is very little that can be done to overcome problems when the actual field value itself has been compromised by poor OCR, but some possible options include:

1) Adjusting scanner settings and/or utilizing image processing technologies, 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

Generic field extraction can also be compromised by poor OCR on the contextual information surrounding the field, or no contextual information at all. The Brainware 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.

In the example below, the context surrounding the invoice number cannot be read properly, hence the confidence value of the field has fallen below the threshold required:

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 following shows an instance where no context at all has been provided for the invoice number (quoted as 'INV-31401'), so, from the system's perspective, the correct value is 'floating in mid-air':

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 which 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 SLW (Supervised Learning Workflow)

The 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.

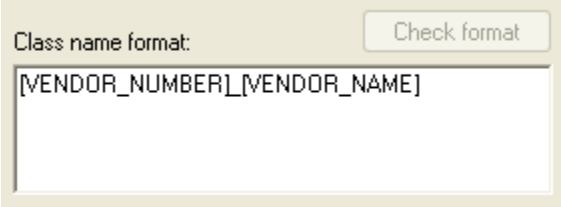
It is appropriate for use in instances where field data is not being extracted due to low system confidence or in instances where the system is delivering an incorrect extraction result. It cannot remedy problems caused by poor OCR, except in instances where the OCR issue concerns the surrounding field 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 the Verifier application. However, if the invoice is from a high-volume vendor, then creating a class makes much more sense.

Oracle recommends a limit of 500 SLW classes per project file.

In a production environment, learnset additions are proposed by AP 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 class which 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, shown below:



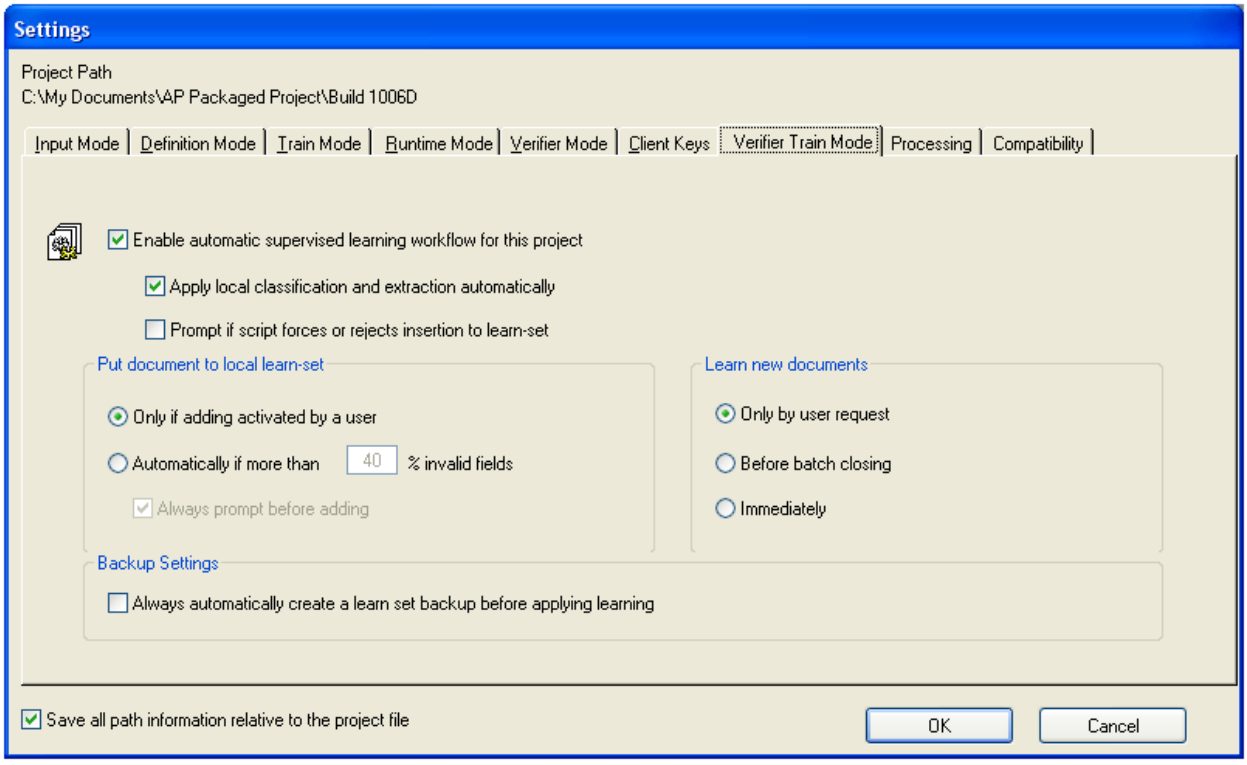
Typically, this is set to the name of the vendor, then an underscore, then the vendor ID, although the class is actually used for any invoice of a similar layout which exhibits a similar extraction problem.

8.2.2.1 Using SLW in Designer

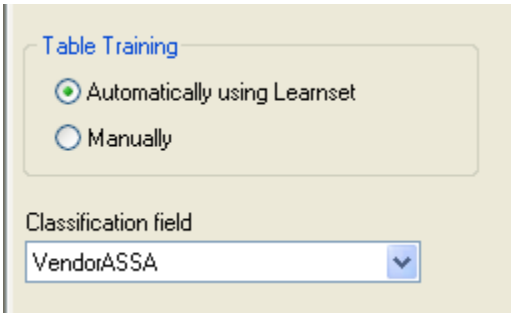
It is possible to use the Supervised Learning Workflow feature within the Designer application, which is done via 'Verifier Train Mode' as a fast-track option. This process is recommended for a system administrator, as it short-cuts the process of having to 'approve' the invoice using the Learnset Manager 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, shown below (recommended settings):

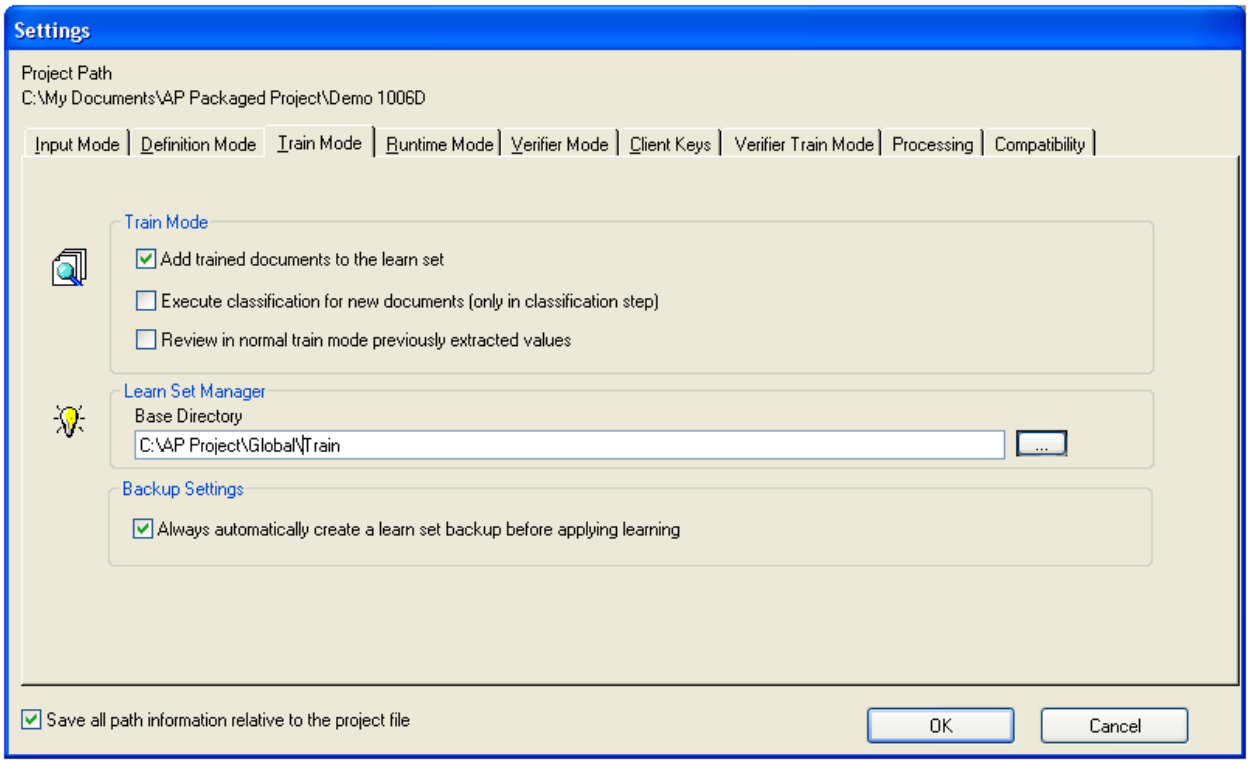


2) The classification field on the 'Document Class' tab of the 'Invoices' class level should be set to 'VendorASSA';

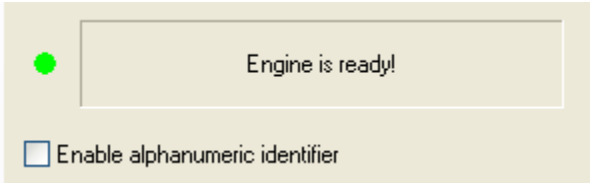


3) A Verifier form must be defined against the 'Invoices' class level;

4) The project should be pointing to the relevant learnset training folder – this can be seen on the 'Train Mode' tab of the project settings:



5) The pool against the 'VendorASSA' field must be set up and showing a 'green light':



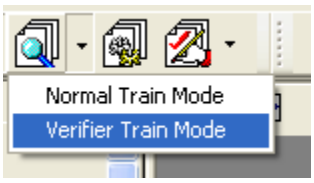
Once these pre-requisites have been fulfilled, it is now possible to train a document.

The steps involved are as follows:

1) Navigate to the problem document in the batch;

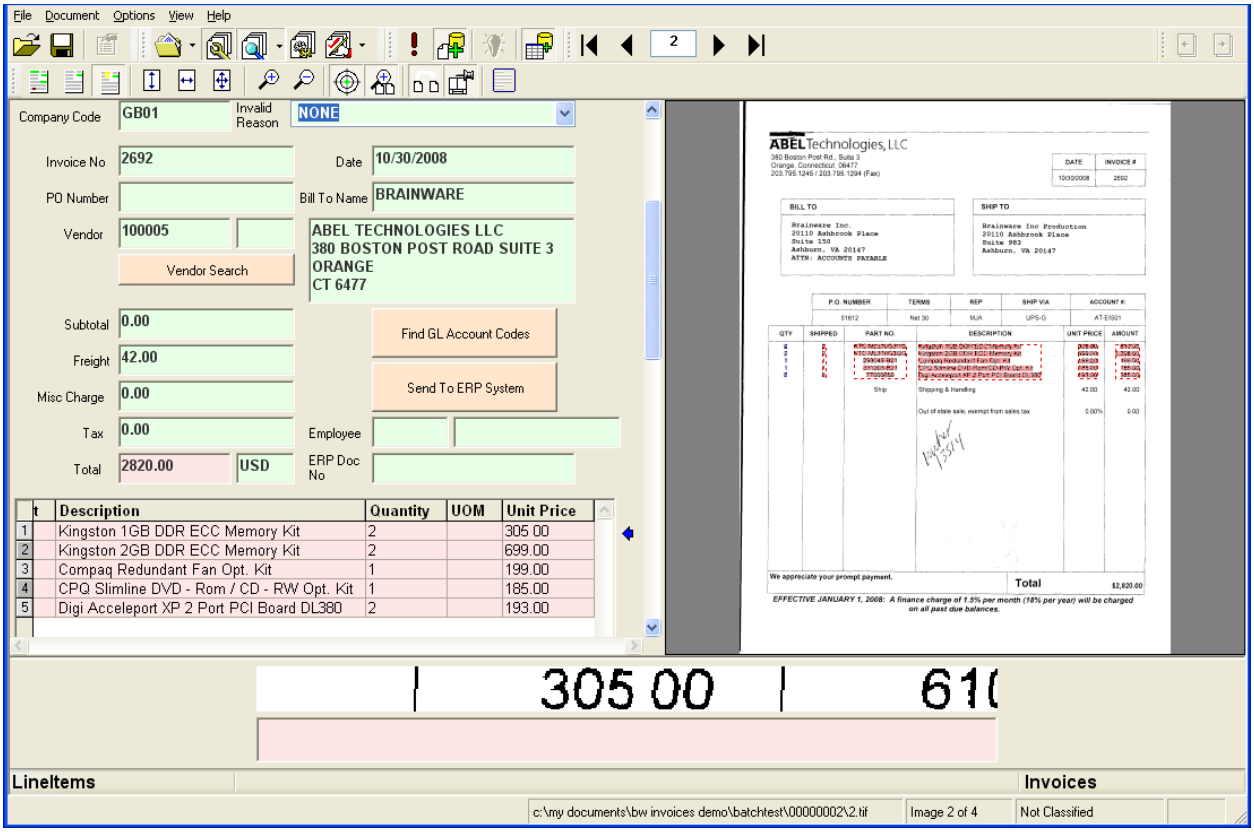
2) Analyze the document in the fields view of ‘Definition Mode’;

3) Select Verifier Train Mode from the top toolbar;



4) Ensure that the ‘Add to learnset’ button  is depressed:

5) Complete verification of the document:

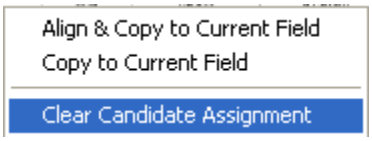


When completing header data, the user must be sure to click on the correct text on the image of the document to copy it into the field so that the system knows where the data was selected on the document. The new class will not be trained properly if the data is simply keyed in manually.

If the document requires line items to be extracted, these must be trained using the Table Extraction 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, hitting ‘enter’ on the last invalid field will move the system to the next document in the batch.

6) Click the bulb button  to train the class;

7) Save the project file.

Navigating back to the class tree in Definition Mode at this stage will show 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 either 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:

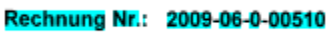
1) Adjusting candidate format strings and definitions;

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 class

In this example, the invoice number stated on the German invoice below is ‘2009-06-0-00510’.



The new class is extracting ‘00510’ (i.e. just the latter part of the number) as the field result.

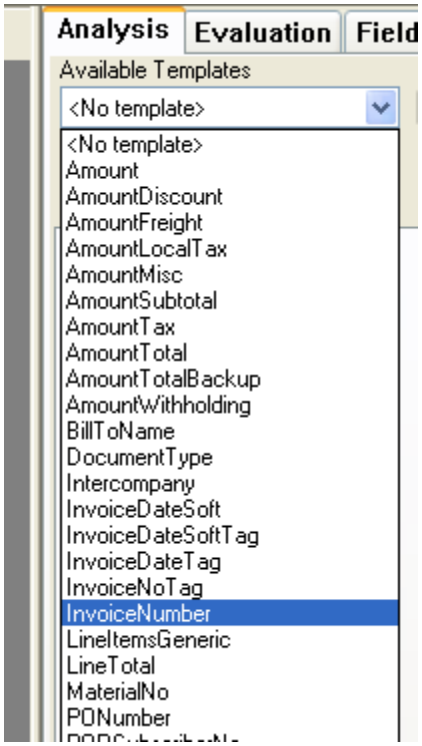
The reason this is occurring is because the format string for the invoice number which the new class has inherited from the field definition on the ‘Invoices’ level is insufficient to extract the invoice number in its entirety.

The corrective course of action is, therefore, to adjust the format settings for the invoice number on the new class.

To do this, go to the class view screen and double click on the new class to see the fields.

Navigate to the invoice number field and show the field properties.

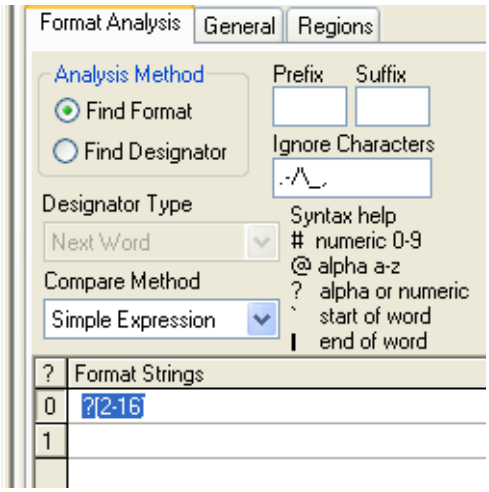
From the drop-down of ‘Available Templates’, select the ‘InvoiceNumber’ template (shown below):



Now click the ‘Copy Template’ button.

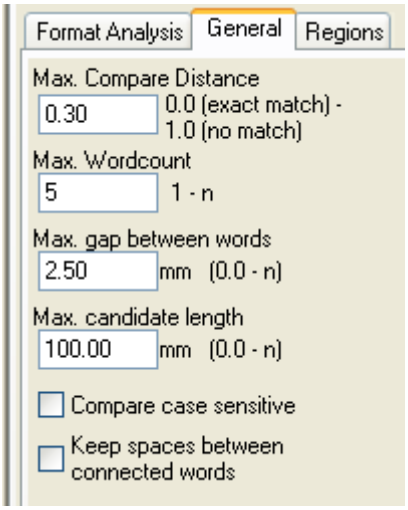
The generic format settings for the invoice number will now be displayed. The user is free to manipulate these settings is any way deemed appropriate as they only apply to the new document class, and will not have a wider effect on the project. Under no circumstances should the format settings against fields on the ‘Invoices’ class ever be changed; doing so may have a catastrophic effect on the generic extraction results.

The standard settings are shown below:



The generic format string is '?[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 of '.-/_,' may also be permitted to appear in an invoice number candidate in any position.

The correct invoice number is 16 characters in length, so these format settings are OK and do not require changing. Now click on the general tab to show the general settings (shown below):

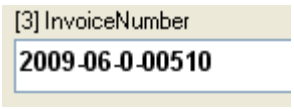


In the candidate definition, the settings above show that the invoice number is permitted to be a maximum of 5 OCR words. On the invoice itself, the desired invoice number consists of 7 OCR words:

i.e. '2009', '-', '06', '-', '0', '-' and '00510'.

Hence, the corrective course of action is to increase the 'Max. Wordcount' property to 7.

With these new settings, the invoice number will now be extracted correctly for the class without jeopardizing the extraction results for any other documents.



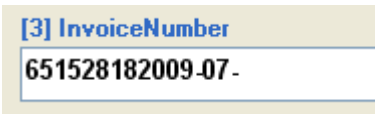
Now save the project.

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 (shown in the 'Max. gap between words' parameter). Increasing this distance can help solve this problem.

Example 2: The system is pulling out part of the date on the end of the invoice number for the new class

In this example, the invoice number and date are presented on the invoice as shown below:

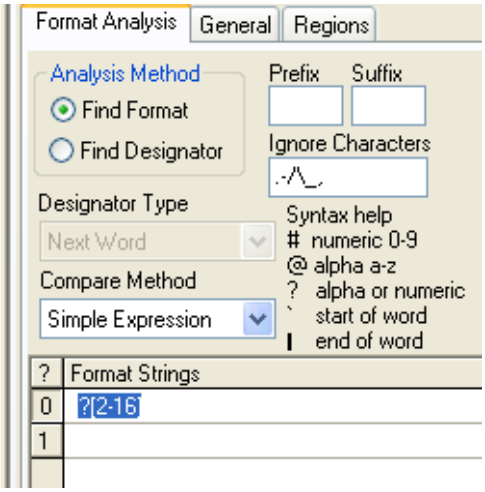
The invoice number has been read into the field as:



In short, part of the invoice date has been added to the end of the invoice number extraction result.

To correct this issue, the format string definition for the invoice number must be amended against the settings for the new class.

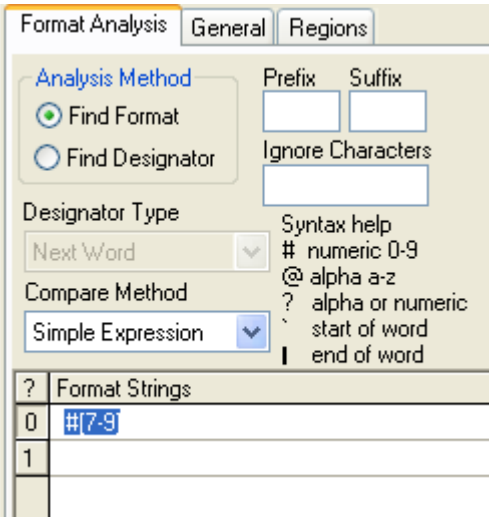
Following the steps in example one, go to the field settings for the invoice number against the new class, select the 'InvoiceNumber' template and copy it down. The following settings will be displayed:



The default generic format string needs to be changed to something more suitable for this type of invoice. The correct invoice number is eight digits in length, so the format string should be changed to something 'tighter', for example #[7-9], which denotes any number between 7 and 9 characters in length.

Make the change to the format string as shown below; as the invoice number for this vendor does not contain any special characters, the 'Ignore Characters' field can be emptied.

The amended settings are shown below:



With this new, more exact format string applied, the invoice number will now be extracted correctly for the class without jeopardizing the extraction results for any other documents.

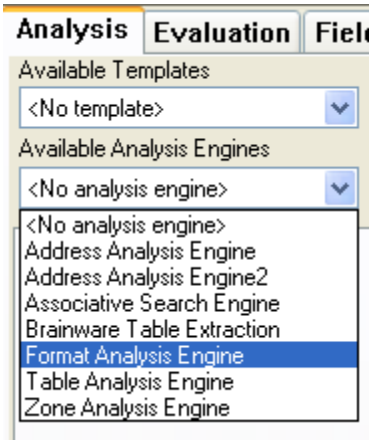
Now save the project.

Example 3: 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.

To fix, navigate to the 'AmountTax' field against the new class and show the field properties.

Select 'Format Analysis Engine' from the list of available engine:



From here, simply configure no format strings. This means that no candidates can ever be generated for the tax amount hence no value can ever be extracted.

Now 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:

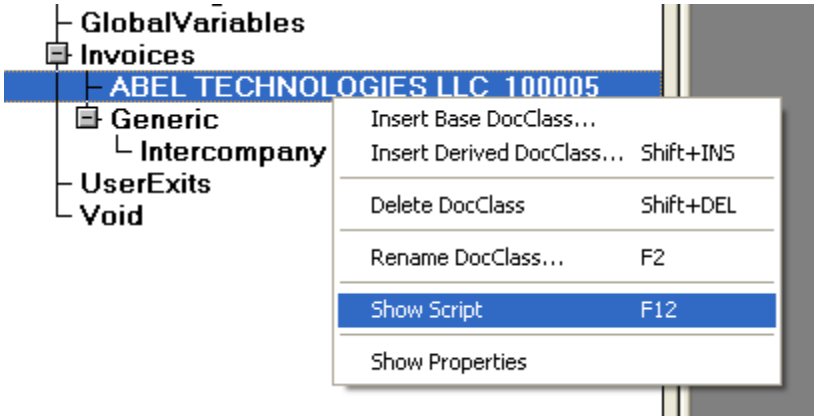
1) Correcting OCR problems where the correct result will always be known for a given vendor;
2) Defaulting mandatory field values which the vendor does not actually state on the invoice;
3) Improving line item extraction for vendors who present the information in a way that is not 100% supported by the Table Extraction engine.

The following sections provide some examples illustrating the above:

Example 1: Script correction of a known OCR issue with the invoice number

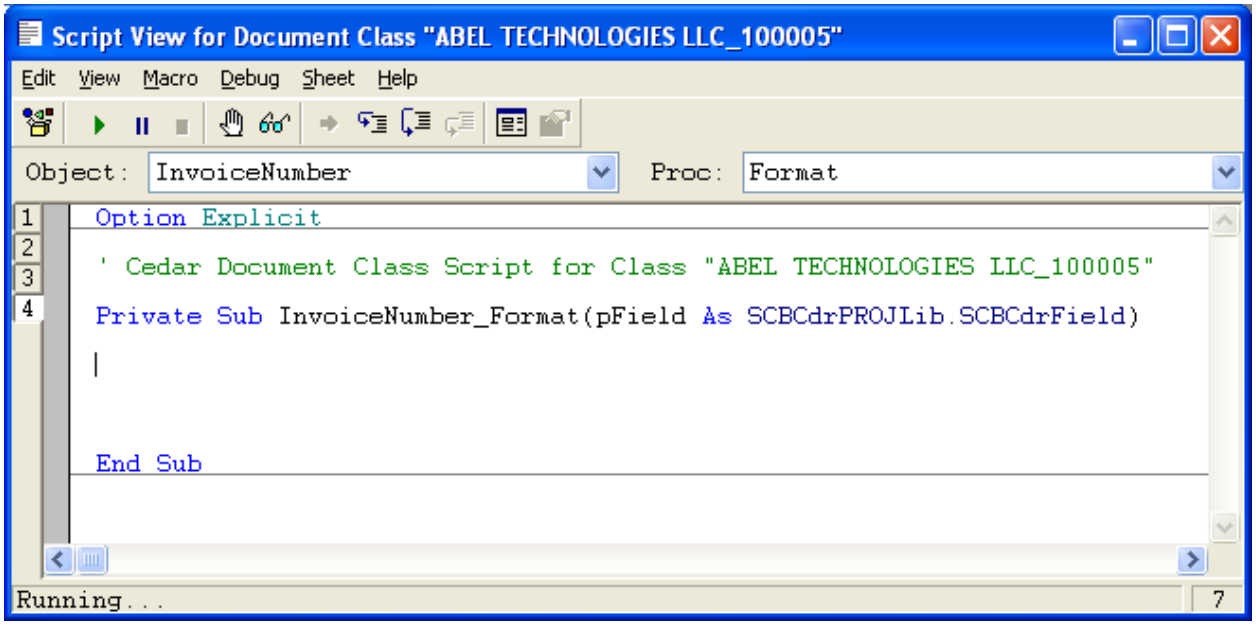
The vendor presents an invoice number that is always suffixed with 'RI'; however, the OCR engine always reads it as 'RL'. As a consequence, documents from this vendor are always stopping in Verifier for user correction.

To correct, go to Designer Definition Mode (classes view), right click on the custom class name and select 'Show Script' from the context menu:

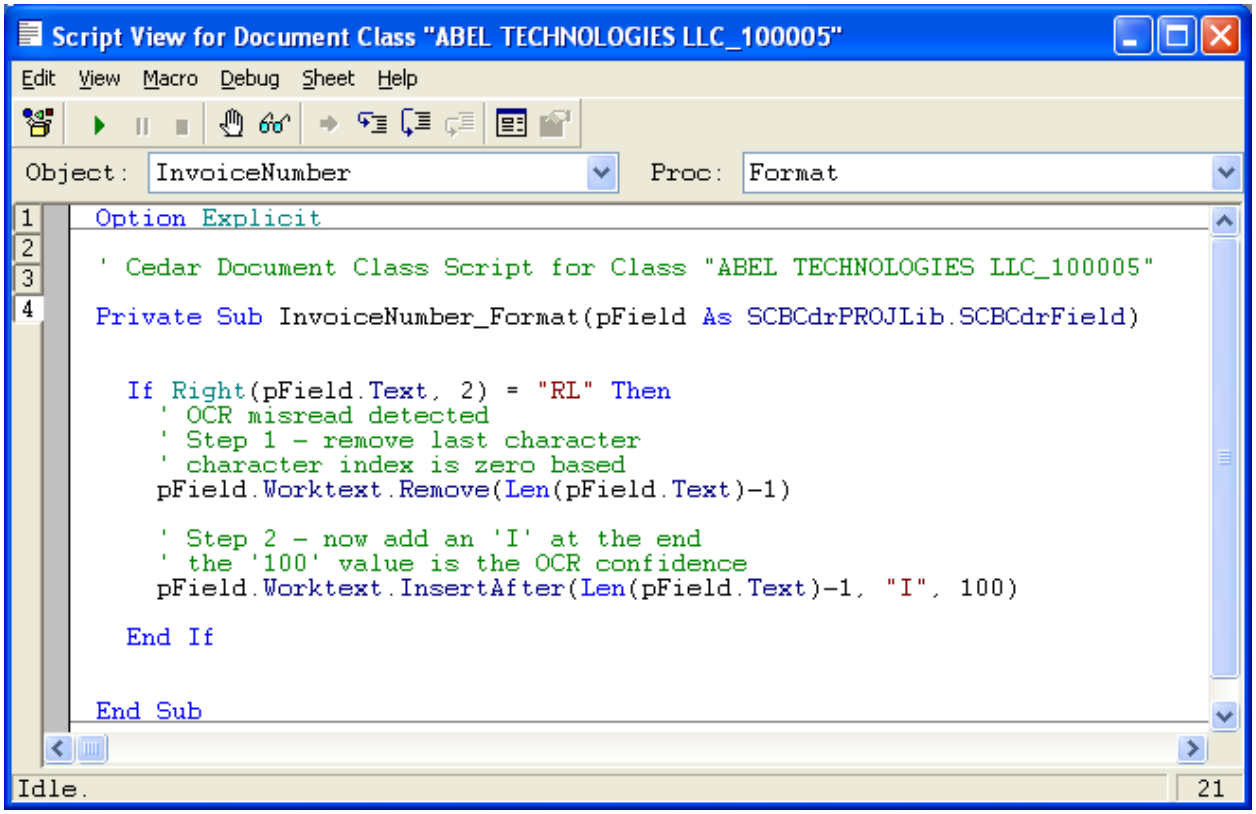


The system will then display the custom script for the 'ABEL TECHNOLOGIES' class.

The correct system event to insert the corrective script for a header field is 'InvoiceNumber_Format'. Selecting 'InvoiceNumber' from the object menu, and 'Format' from the procedure menu will automatically create the subroutine header and footer, as shown below:



Now, code can be inserted into the subroutine in order to clean up the OCR misread. The corrective code (which maintains the OCR confidence values of other characters in the string) is shown in the screenshot below:



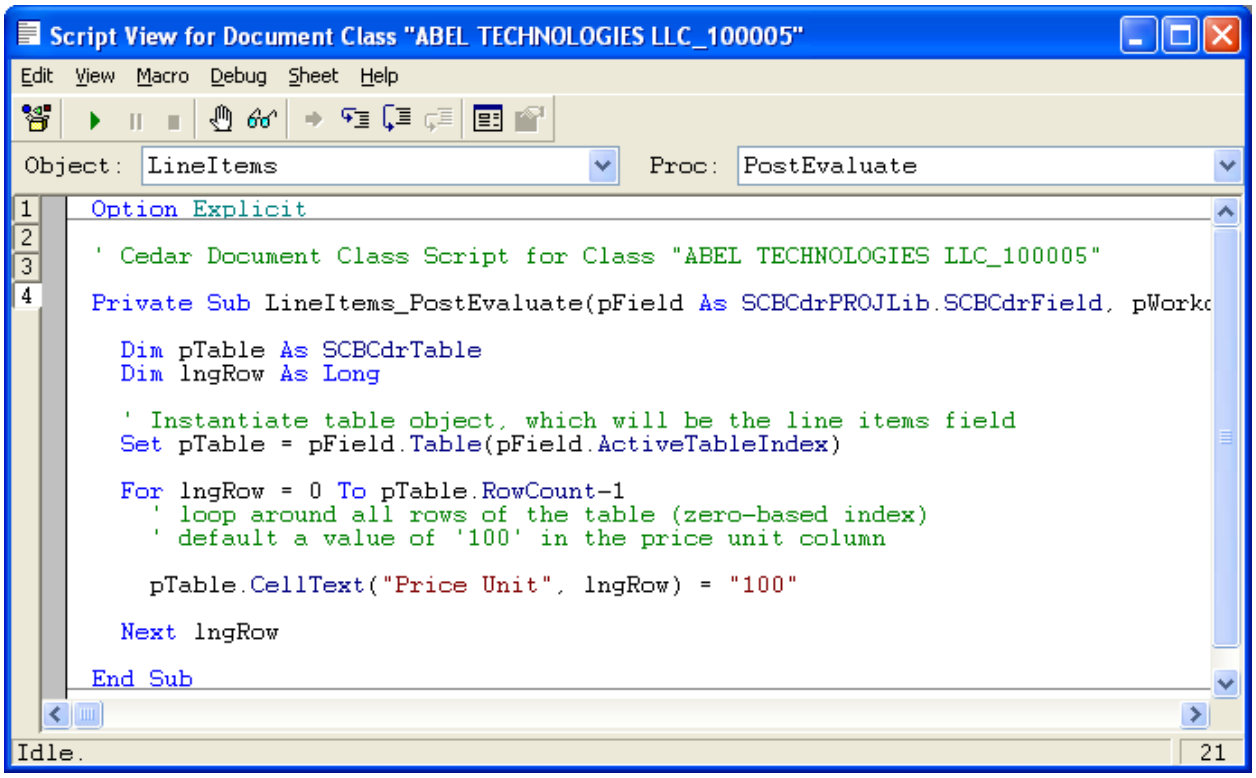
For complete over-writes of the invoice number where the 'Remove' and 'InsertAfter' methods are too cumbersome to employ, the text can be replaced in its entirety using the following command:

```
pField.Worktext.Text = "My Value"
```

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.

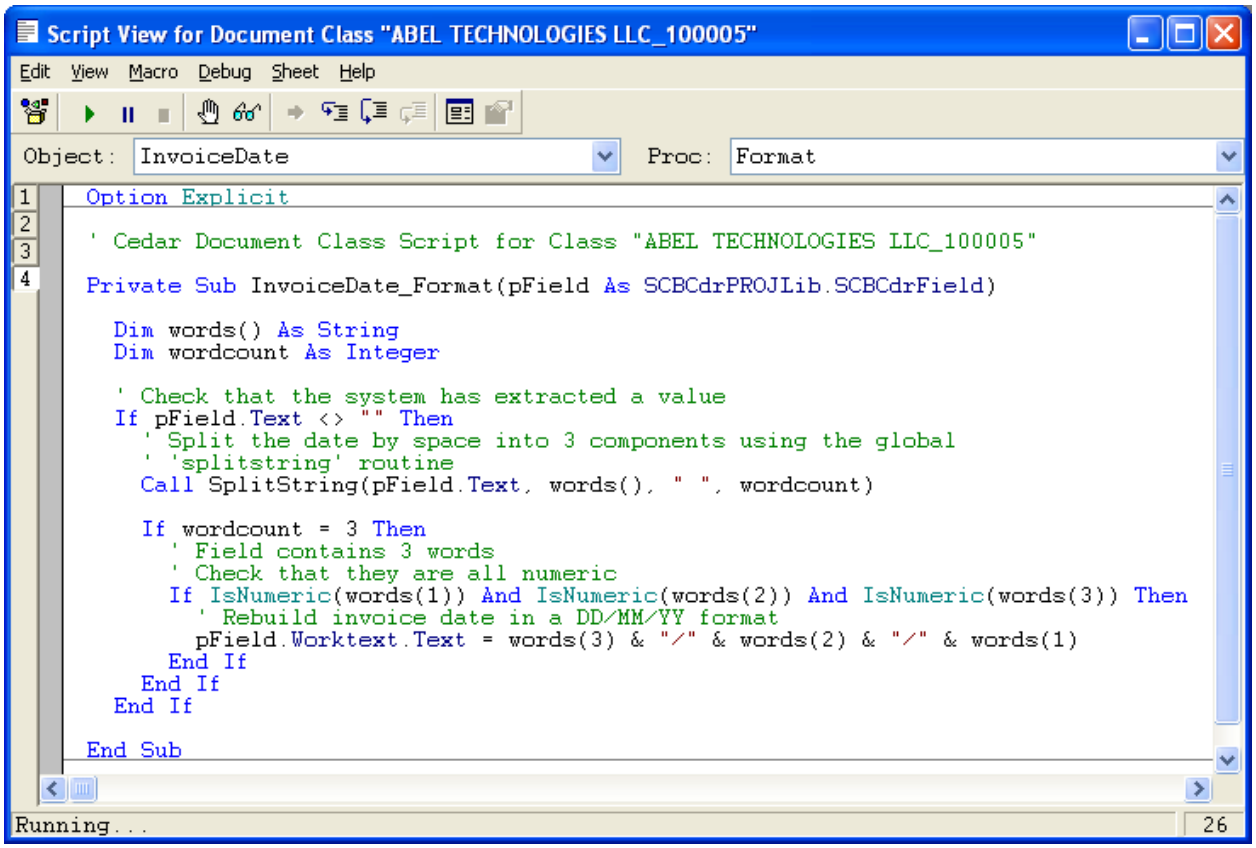
To handle this issue, custom script needs to be inserted into the 'LineItems_PostEvaluate' event against the class. This code is shown in the screenshot below:



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.

The screenshot below shows custom script that can be inserted for the vendor class in order to automate this. As it concerns a header field, this script belongs in the 'InvoiceDate_Format' event:


Script View for Document Class "ABEL TECHNOLOGIES LLC_100005"
Edit View Macro Debug Sheet Help
Object: InvoiceDate
Proc: Format
Option Explicit
' Cedar Document Class Script for Class "ABEL TECHNOLOGIES LLC_100005"
Private Sub InvoiceDate_Format(pField As SCBCdrPROJLib.SCBCdrField)
Dim words() As String
Dim wordcount As Integer
' Check that the system has extracted a value
If pField.Text <> "" Then
' Split the date by space into 3 components using the global
' 'splitstring' routine
Call SplitString(pField.Text, words(), " ", wordcount)
If wordcount = 3 Then
' Field contains 3 words
' Check that they are all numeric
If IsNumeric(words(1)) And IsNumeric(words(2)) And IsNumeric(words(3)) Then
' Rebuild invoice date in a DD/MM/YY format
pField.Worktext.Text = words(3) & "/" & words(2) & "/" & words(1)
End If
End If
End If
End Sub
Running...
26

APPENDIX A: TAX CONFIGURATION FOR COUNTRIES WITHOUT TAX JURISDICTIONS

A1 Introduction to the Tax Table

Oracle Forms Recognition uses a look-up table for the purposes of determining and validating tax codes for purchase order based invoice 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 Oracle will provide. Usage of the tax table is activated in the TAX section of the system configuration (see section 4.1.12).

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 either to the country of the company which 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 '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.
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
Percentage	Number (two decimal places)	Percentage rate associated with the tax code (For example, 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, Oracle 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 Tax Code Validation

Once tax codes for all invoice lines have been determined, the system uses the percentage rate against each tax code (except in cases of foreign tax being charged) to compare what the tax should be using those codes against what has actually been billed on the invoice. If the amounts are outside of the configured tolerance, the invoice will go to exception.

If any single invoice line has been allotted a foreign tax code, which is determined by the ship-from country differing from the company code country, then the entire tax amount on the invoice is booked to that tax code.

If the tax table contains no entries for the country of the invoice company code, no validation is carried out.

A5 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	TaxCode	VendorID	ShipTo	ShipFrom	Service	Material Group	MaterialNo	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
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) which 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	TaxCode	VendorID	ShipTo	ShipFrom	Service	Material Group	MaterialNo	Percentage
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	TaxCode	VendorID	ShipTo	ShipFrom	Service	Material Group	MaterialNo	Percentage
GB	DS		GB	GB	X			999

A6 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.

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	TaxCode	VendorID	ShipTo	ShipFrom	Service	Material Group	MaterialNo	Percentage
GB	VR	0000012345	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 (for example, 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 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 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 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 in the system configuration (see section 4.1.12):

```
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=NO
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. A blank account assignment category is denoted by an asterisk.

The new purchase order tax code selected by the process above is then subject to the logic described in section C2 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’ and ‘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 system 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 system 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, SRC and IMP sections of the system configuration:

Business Requirement 1: The invoice type should be NO-PO unless a purchase order is found on the document.

ITY Section:

```
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
```

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 two parameters should be set as follows:

```
ITY_OP_SetToPOIfPOFound=NO
ITY_OP_SetToPOIfValidPOFound=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 Section:

```
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
```

SRC Section:

```
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 Section:

```
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
```

IMP Section:

```
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 Oracle Forms Recognition. 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 alpha-numeric.

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.

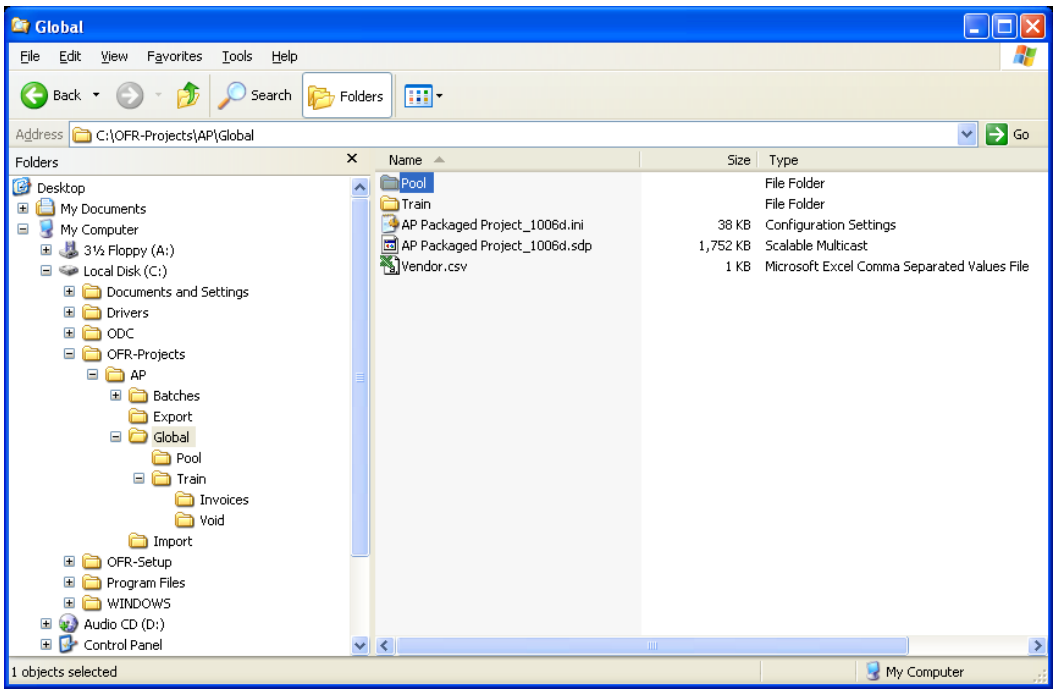
D1 Configuring a Standard Vendor ID

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 client, whether it is provided as a CSV file or within a database table, 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. The steps are as follows:

1) In the project file directory, create a pool directory as shown below:



2) Open the .ini file 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=\\MyComputer\Projects\Global\Pool
ASA_VL_01_PoolDirectory=Pool
ASA_VL_01_PoolName=Vendor
ASA_OP_01_FileRelative=NO
ASA_VL_01_ImportPathFilename=\\MyComputer\Projects\Global\vendor.csv
ASA_VL_01_ImportFilename=vendor.csv
```

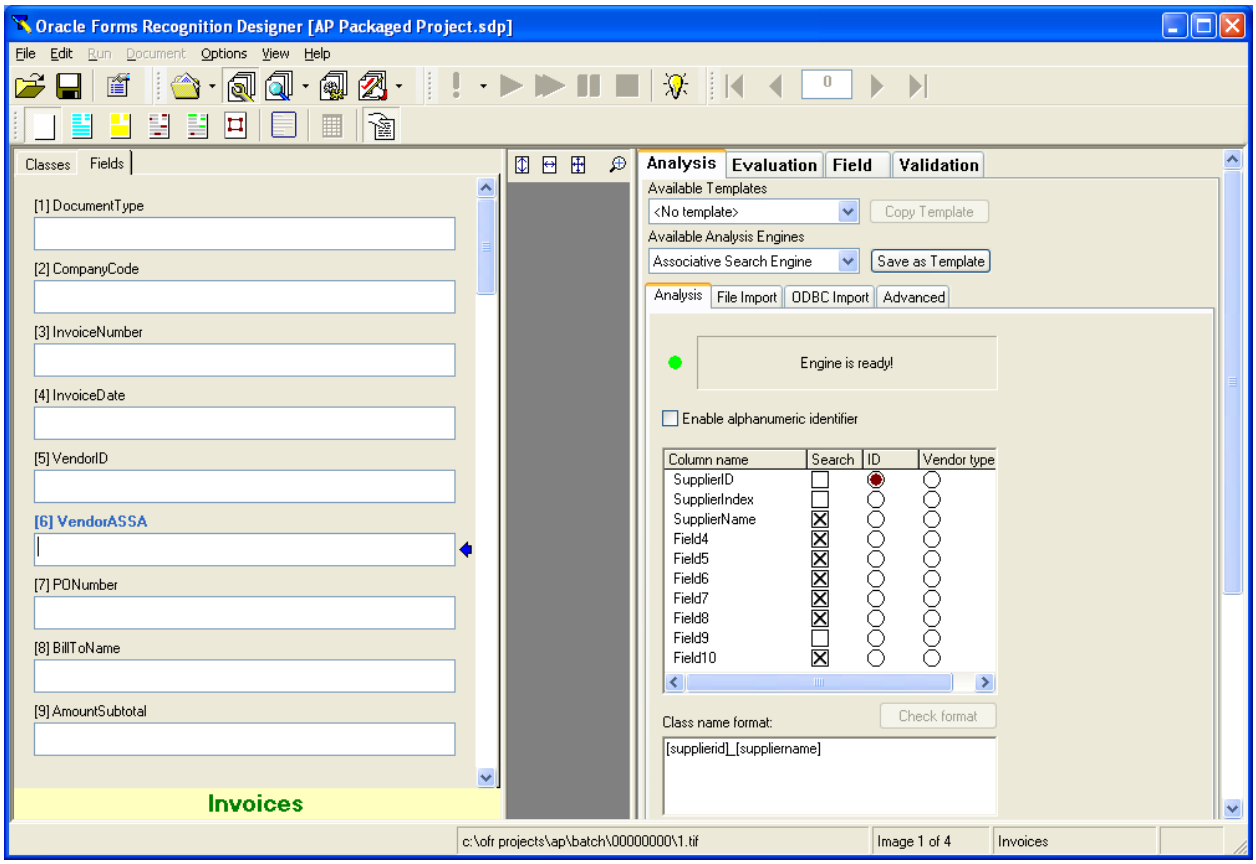
If the unique identifier for each row in the vendor extract is numeric, then the 'AlphaNum' parameter should be set to 'NO'. In all other cases, it should be set to 'YES'.

If supervised learning is to be used for the client deployment, 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' can be left empty, 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.

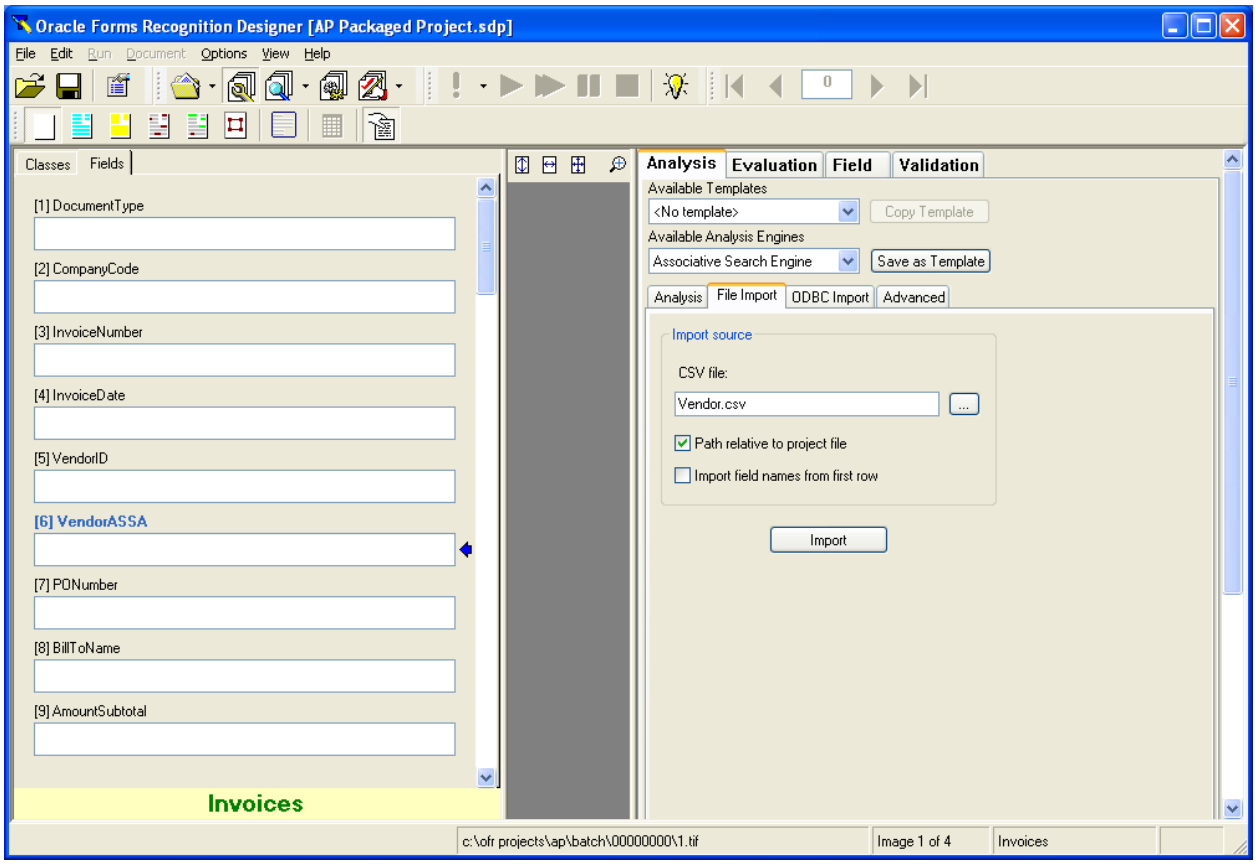
If the vendor extract file is not to be placed in the same directory as the project file, then the 'FileRelative' parameter must be set to 'NO', and the 'ImportPathFileName' must be populated with the UNC path to the vendor extract file. If the vendor extract file is located in the same directory as the project file, the 'FileRelative' parameter can be set to 'YES' and the 'ImportFileName' must be populated with the name of the vendor extract file (including the file extension).

3) Open the project file, and navigate to the 'VendorASSA' field on the invoices class. Show the field settings.



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. 'SupplierID' will always be the first column in the extract; 'SupplierIndex' the second; 'SupplierName' the third; 'Field4' the fourth and so on…

4) Go to the file import tab and press the 'Import' button to import the pool:



5) Move back to the ‘Analysis’ 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, the street address, the city, the zip/postal code and, if appropriate, the vendor telephone numbers and tax/VAT identifiers.

6) Now use the radio button in the ID column to select the column in the vendor extract that denotes the unique identifier for the vendor record;

7) Go back to the ‘File import’ tab and re-import the pool;

8) Configure the class settings. This should always be set to:

   ```
   [*vendor name*] + underscore + [*vendor ID*]
   ```

   If the column marked ‘SupplierID’ denotes the unique vendor ID and the column marked ‘SupplierName’ denotes the vendor name (which will vary on the column order in the actual vendor extract file itself’, then the correct entry in this field would be:

   ```
   [SupplierName]_[SupplierID]
   ```

   The entry in the class settings box denotes 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 client or not.

9) Configure the field settings. The field settings control 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.

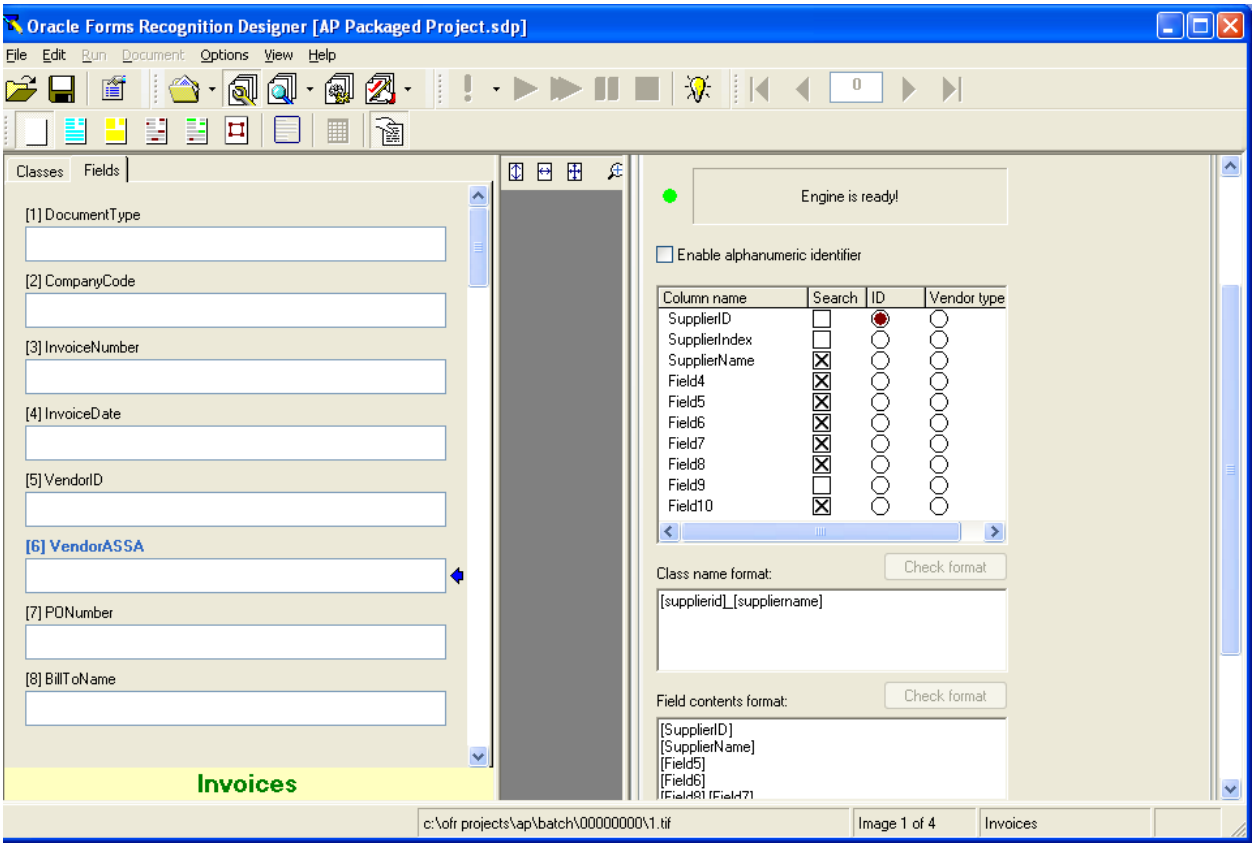
It is recommended that the field follows the following structure, but this is optional depending on what is appropriate for the client.

```
[*Unique ID*]
[*Vendor Name*]
[*Street Address*]
[*City*]
[*State / Region*] [*Postal / Zip Code*]
```

In an example where SupplierID represent the unique record identifier, SupplierName represents the vendor name, Field4 represents the vendor street address, Field6 represents the vendor city. Field8 represents the vendor state and Field7 represents the vendor zip / postal code, the configuration should be as follows:

```
[SupplierID]
[SupplierName]
[Field4]
[Field6]
[Field8] [Field7]
```

The screenshot below shows completed configuration for the above example:



10) At this point, the vendor field configuration is complete so a green light with the message 'Engine Is Ready' should appear in the field status box. Save and close the project file.

11) 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 system 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=SupplierID
SRC_VL_SiteID=
SRC_VL_Name=SupplierName
SRC_VL_Address1=Field5
SRC_VL_Address2=
SRC_VL_City=Field6
SRC_VL_Zip=Field7
SRC_VL_State=Field8
SRC_VL_Country=Field9
```

    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 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. Oracle Forms Recognition does permit use of a composite key within the vendor extract, so the two values will need to be brought together within a single column.

The rules 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 * 1000000) + 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] + Site ID

For example: if the vendor ID is A12345, the Site ID is 1000, and the designated separator is a hyphen (-), then the vendor extract should have ‘A12345-1000’ in the unique identifier column.

The nominated separator is specified under the parameter ‘AlphNumSiteSeparator’ in the VND section of the system configuration. This MUST be populated and adhered to if alpha-numeric vendor and site IDs are to be used. The system will throw an error if this has not been done correctly, or more than one separator is found as part of a single unique identifier.

Hence, in the extract file, three columns are required: one representing the vendor ID; one representing the site ID; and one representing the combined unique identifier.

To configure the vendor field, all of the steps specified in Section D1 should be followed, with two variations:

1) For step 8, the ID used in the class-name should not be set to the field that represents the unique identifier for the vendor record, but rather the field that represents the standalone vendor ID;

2) For step 11, in the SRC section of the system configuration, the ‘ID’, ‘SiteID’ and ‘ExternalVendorID’ parameters must all be mapped.

   The unique record 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.

   For example: if Oracle Forms Recognition has allocated technical names of ‘SupplierID’ to the unique ID column, ‘SupplierIndex’ to the site ID, and ‘Field11’ to the vendor ID component, then the mappings should be as follows:

```
SRC_VL_ID=SupplierID
SRC_VL_SiteID=SupplierIndex
…
SRC_VL_ExternalVendorID=Field11
```

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 client 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.

To carry out this configuration, all the steps described in Section D1 should be followed with two variations:

1) For step 8, the ID used in the class-name should not be set to the field that represents the unique identifier for the vendor record, but rather the field that represents the external vendor ID;

2) For step 11, in the SRC section of the system configuration, the 'ID' and the 'ExternalVendorID' parameters must be mapped.

   The unique record identifier should be mapped to the 'ID' parameter, and the external vendor ID should be mapped to the 'ExternalVendorID' parameter.

   For example: if Oracle Forms Recognition has allocated technical names of 'SupplierID' to the unique ID column, and 'Field11' to the external vendor ID component, then the mappings should be as follows:

```
SRC_VL_ID=SupplierID
…
SRC_VL_ExternalVendorID=Field11
```

If the downstream uses both an external vendor ID AND a site ID, then the SRC section should be mapped as follows:

```
SRC_VL_ID=SupplierID
SRC_VL_SiteID=SupplierIndex
…
SRC_VL_ExternalVendorID=Field11
```

In this example, 'Field11' represents the external vendor ID field, rather than the internal vendor ID shown in the example in Section D2.

APPENDIX E: CONFIGURING THE MISCELLANEOUS CHARGES

E1 Introduction

Miscellaneous charges refers 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 A/P Solution provides two fields for this purpose: AmountFreightPrepaidAndAdded and AmountMisc.

At the line item level, the category column is used to identify and classify any miscellaneous charges captured that the vendor specifies as a line item on the invoice. This category is set automatically by Oracle Forms Recognition drawing on the system configuration settings applied in the MSC section (see section 4.1.18).

The handling of miscellaneous charges is dependent on the business rules of the client. The configuration options within the MSC section permit:

1) Miscellaneous charges to be booked as 'unplanned delivery costs' (i.e. a single, header-level amount field within the downstream ERP system).
2) Miscellaneous charges to be booked as a separate invoice line item with a specific line type (Oracle Financials).
3) 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. Hence, line pairing must be activated in the LPR section of the system configuration (see section 4.1.26), 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.

E2 Miscellaneous Charge Categories

Each group within the MSC section of the system configuration represents a miscellaneous charge category. The category denotes the type of miscellaneous charge (for example, freight, a customs charge etc.).

The number of categories that should be configured is dependent on how the client wishes to process these charges.

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 client wished to book specific miscellaneous charges to a specific general ledger account depending on what type of charge it was (for example, account 1000 for freight; account 2000 for customs charges; account 3000 for pallet charges etc…), 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 system configuration is copied into the category column within the line items field.

E3 Assigning Header Fields to a Miscellaneous Charge Category

Each miscellaneous charge category can be assigned fields either at the header or line item level.

In the MSC section of the system configuration (see section 4.1.18), the parameter 'NN_HeaderField' controls the mapping between a 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
```

The name of the 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 keywords should be entered that 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
```

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 (for example, 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:

1) A general ledger account code representing the type of expense
2) A cost object (for example, a cost center, an internal order, a profit center, a project etc…)
3) A tax code

To determine the general ledger account code, the system will firstly look into a custom database table (definition to be provided by Oracle, but population is required by the client). 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 look-up 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 lifted 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/profit center in 'NN_DefaultCostCenter' / 'NN_DefaultProfitCenter' 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 lifted 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 which will have the effect of passing a blank tax code and a blank tax jurisdiction code downstream.

If the custom table look-up is activated, but communication does not succeed either due to missing/incorrect configuration or database unavailability, then the export event for the document will fail.

A sample configuration for posting miscellaneous charges as general ledger entries is 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, then 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 look-up 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.

An example for populating the table for company code GB01, plant 1000, line type ‘A’ and miscellaneous charge category ‘F’ can be found below:

Company Code	Category	Line Type	Plant	GLAccount	CostCenter	Profit Center	TaxCode
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 should be considered when populating the table.

E9 Third Party Freight

Within Oracle Forms Recognition, 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.

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

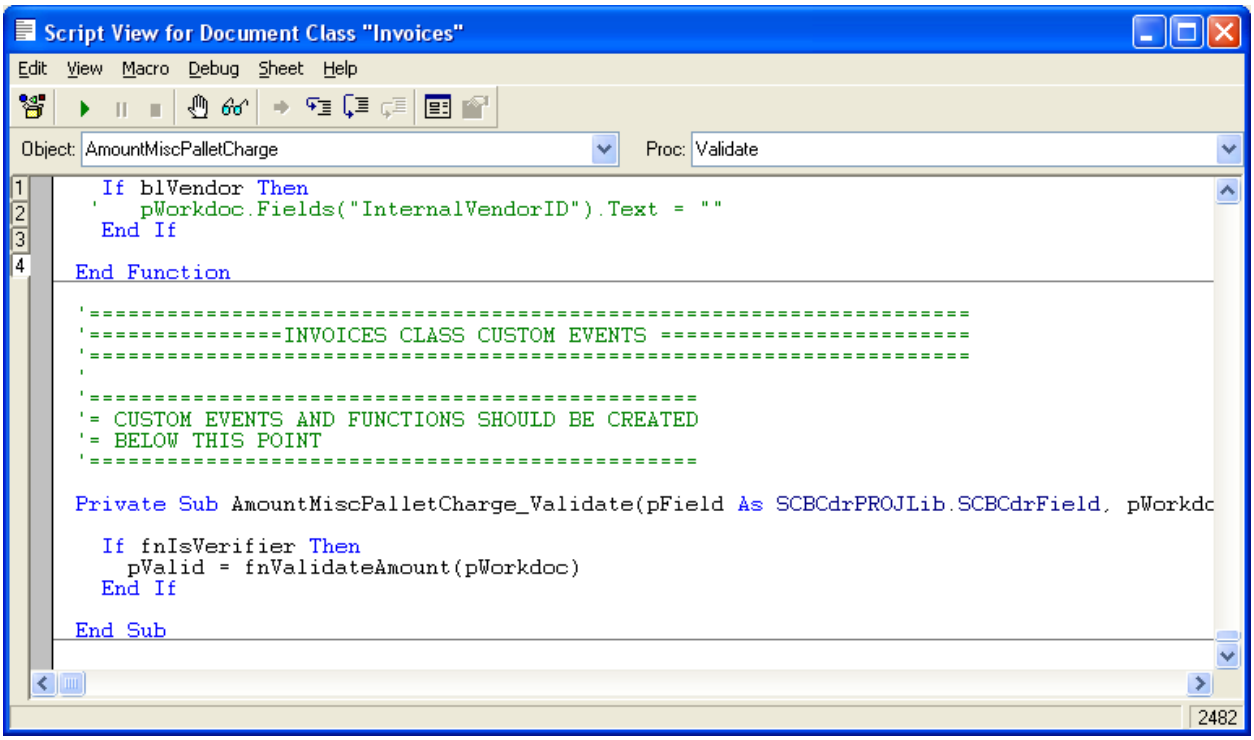
The Oracle Forms Recognition A/P Solution comes bundled with two fields for capturing miscellaneous charges at the header level, specifically 'AmountFreightPrepaidAndAdded' and 'AmountMisc'.

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). 'UserExitAmountMiscPostEvaluate' 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' level, and should be called 'AmountMiscXXX' 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'.

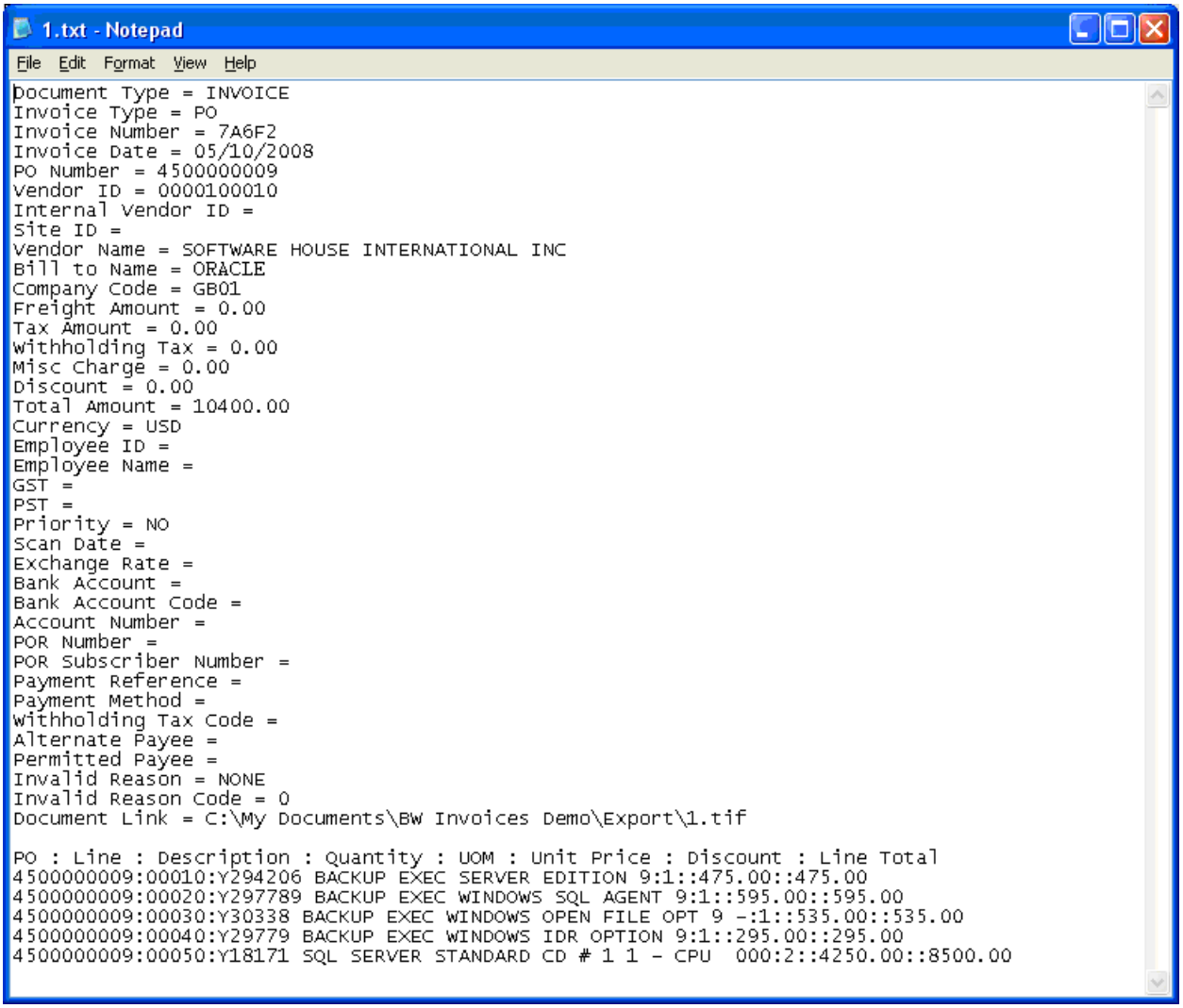
This function will also take care of the field formatting. In the example screenshot below, the new field added has been called ‘AmountMiscPalletCharge’:



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: SAMPLE EXPORT FILES

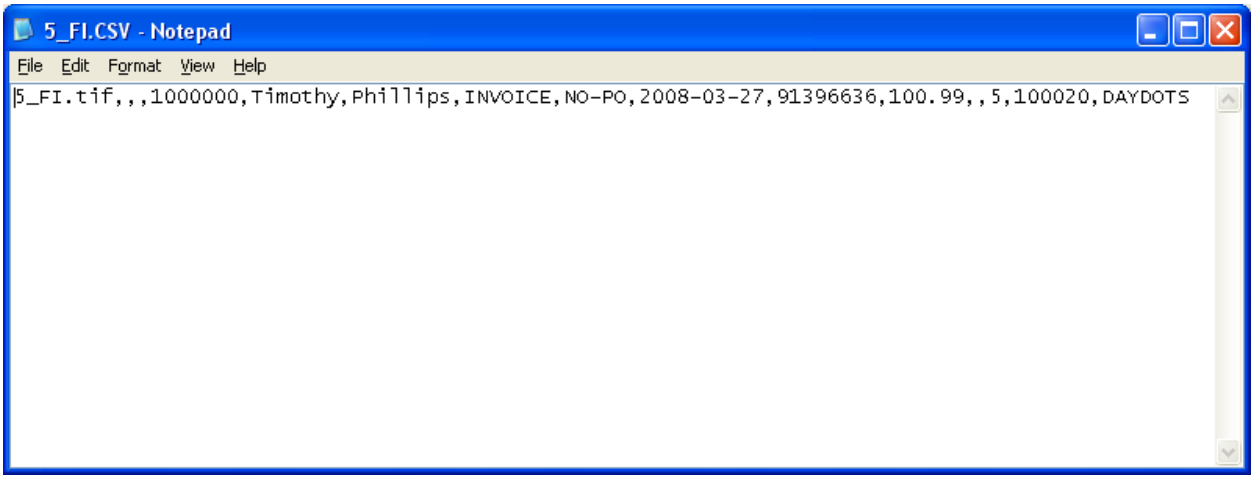
The following screenshot shows an example of the standard export flat file produced during the Oracle Forms Recognition A/P Solution export event:



The options available in the EXP section of the system configuration (see section 4.1.25) control:

1) The file extension used
2) The name of the file (either the entire document image filename, or just the URN component)

The following screenshot shows an example of a CSV output file:

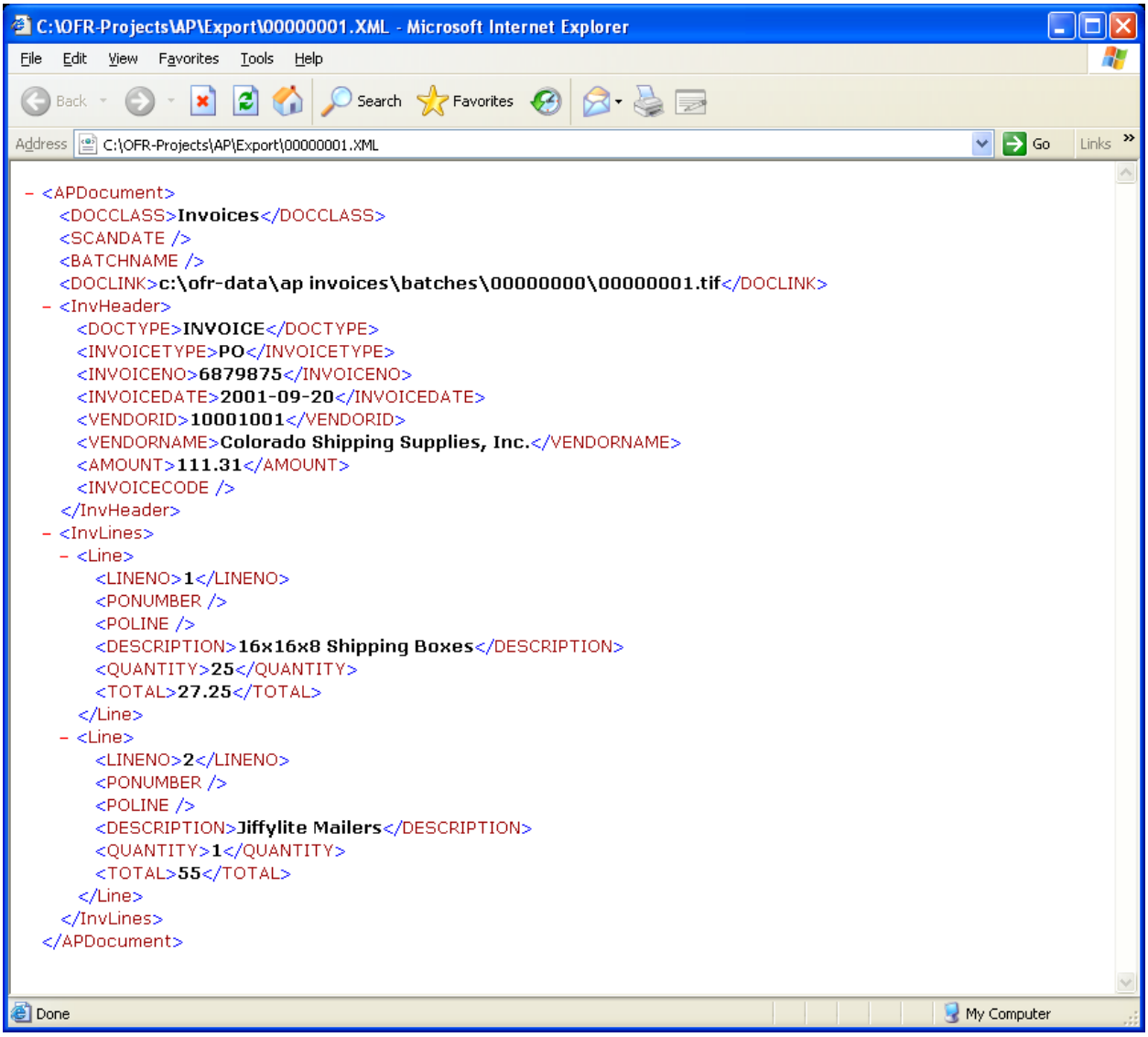


The options available in the CSV section of the system configuration (see section 4.1.30) control:

1) The destination output directory
2) Whether one file should be written per document or per batch
3) Whether the file should be written out for PO or NON-PO invoices or both
4) The file extension used
5) The name of the file (either the entire document image filename, or just the URN component), along with an optional file prefix
6) Whether a copy of the original image should also be written to the output directory
7) The format of any date fields within the file with an optional separator
8) The fields outputted and the order of those fields in the file to a maximum of five lines
9) Whether line items should be outputted and what line item detail should be included
10) The delimiter used to separate the fields

For both files, the system will output the files to the export directory specified in the Runtime Server instance that carries out the export. If no export directory is specified, the export directory specified in the EXP section of the system configuration will be used.

The following screenshot shows a sample XML file output:



The options available in the EXP section of the system configuration (see section 4.1.25) control:

1) The name of the file (either the entire document image filename, or just the URN component).
2) The file extension to be used.
3) All XML tags used within the file.
4) The format of any date fields within the file with an optional separator.
5) Which fields are to be outputted, and which ones are not.

APPENDIX G: DEACTIVATING ASSA FIELDS

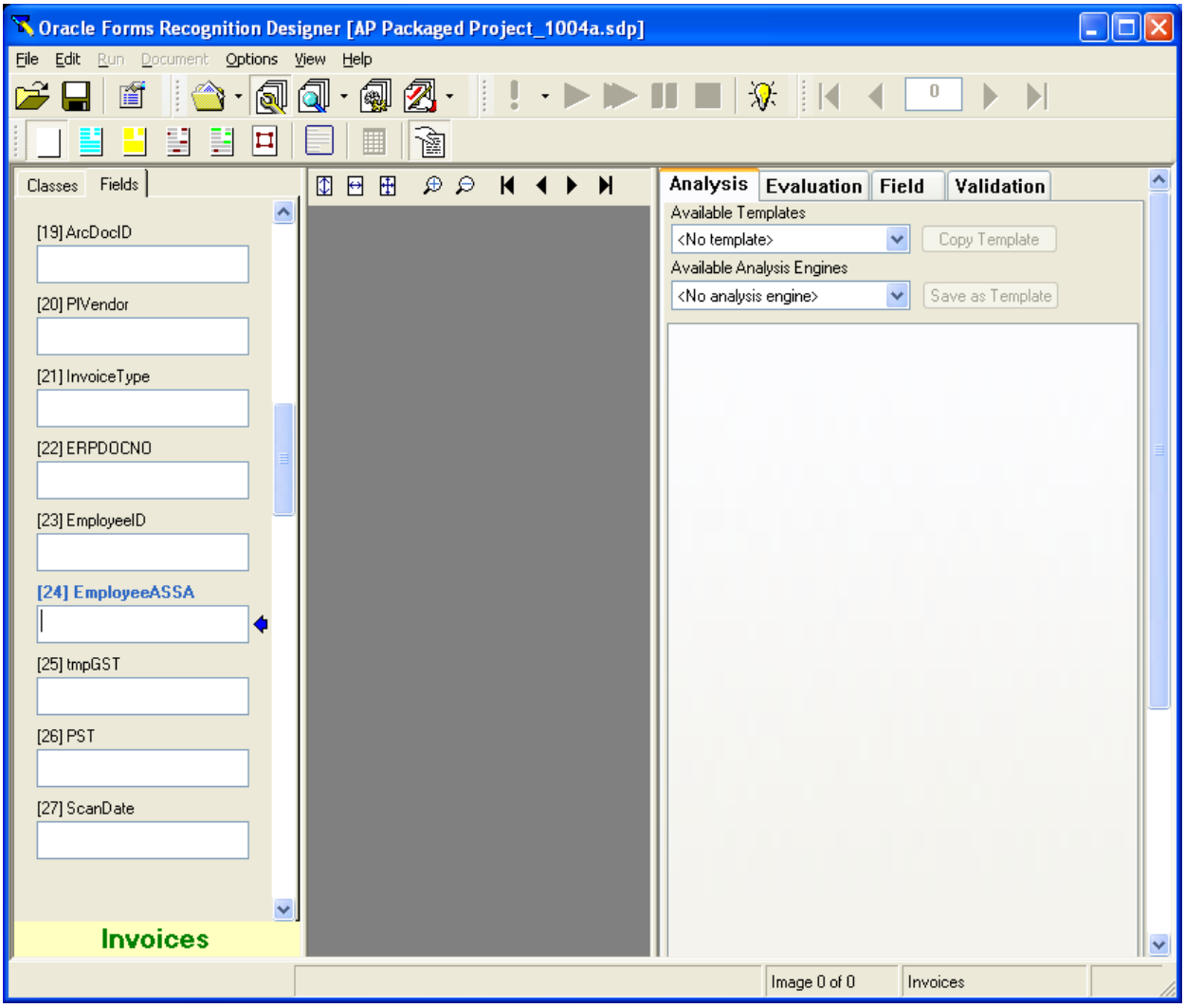
This section describes the steps required to deactivate ASSA fields within the packaged project.

G1 Switching Off the Associative Search Engine

Open the project file using the Oracle Forms Recognition Designer module.

Navigate to the class which holds the definition of the field, then to the field itself.

Show the field properties:



Now, select 'No Analysis Engine' from the Available Analysis Engines dropdown.

Save the project file and close.

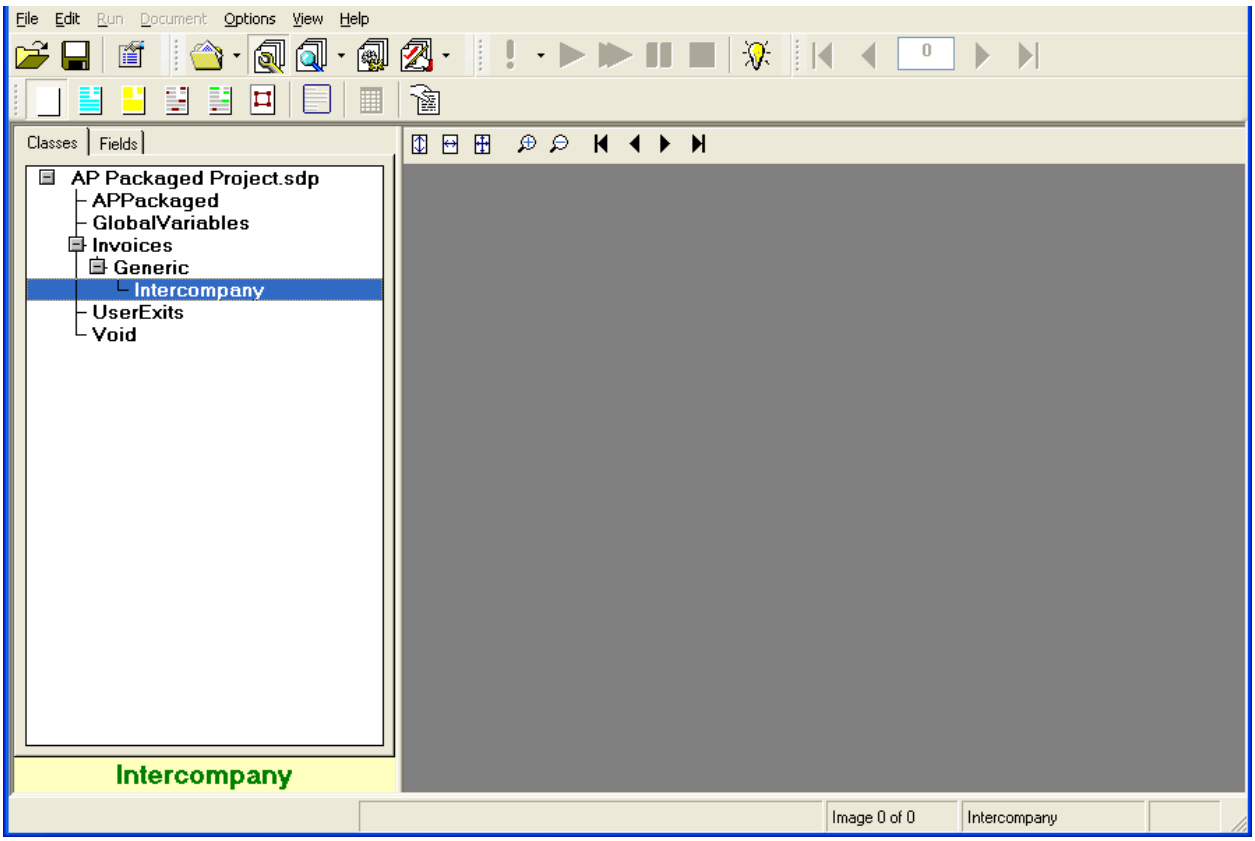
APPENDIX H: HANDLING INTERCOMPANY VENDORS

H1 Introduction

Intercompany invoices arise in instances where different subsidiaries of the same group of companies are invoicing one another. For example: in instances 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 A/P Solution 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 A/P Solution provides a dedicated Intercompany class for this purpose. This class is shown in the screenshot below:



H2 Vendor Extracts

If intercompany invoices are in scope for the solution deployment, then it is incumbent on the client 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 (described in section 6.4 and Appendix E). The intercompany vendor extract file is set up in the same way except that it is mapped to the VendorASSA field held on the intercompany class level.

H3 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 re-classify each intercompany invoice manually.

This may be acceptable if the volume of intercompany invoices is particularly low (i.e. a handful of invoices per month), but, if it is not, then 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.

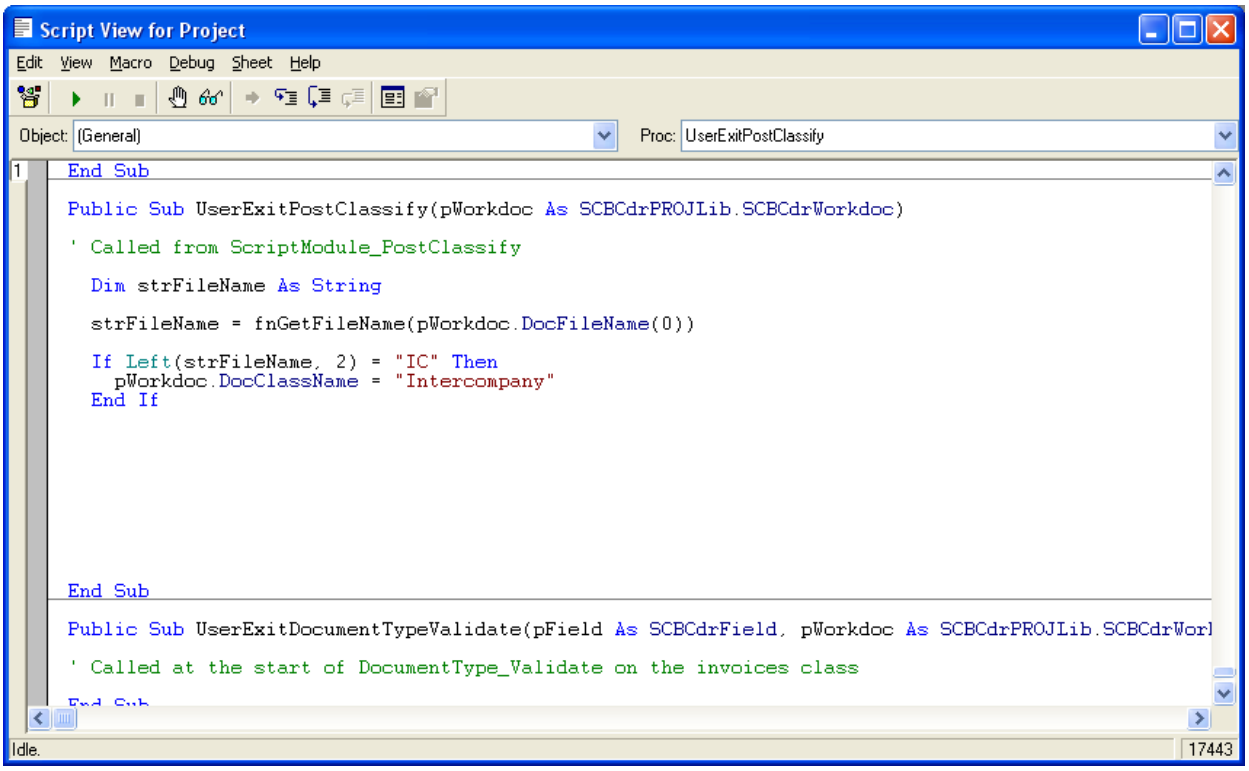
A simple example of this would be if the intercompany scan-job outputs a tiff image with an element built into the filename that denotes that it is an intercompany document.

For example, the first two characters of the tiff filename are 'IC'.

If this is the case, then a simple piece of script in 'UserExitPostClassify' can be used to force the document to the intercompany node, thus guaranteeing 100% correct classification to the extent

that the document was scanned correctly in the first place. In the event of a mis-scan, the document will be classified to the 'Invoices' class and a user will have to re-classify it to the intercompany class manually.

A sample script for the above scenario is shown in the screenshot below:

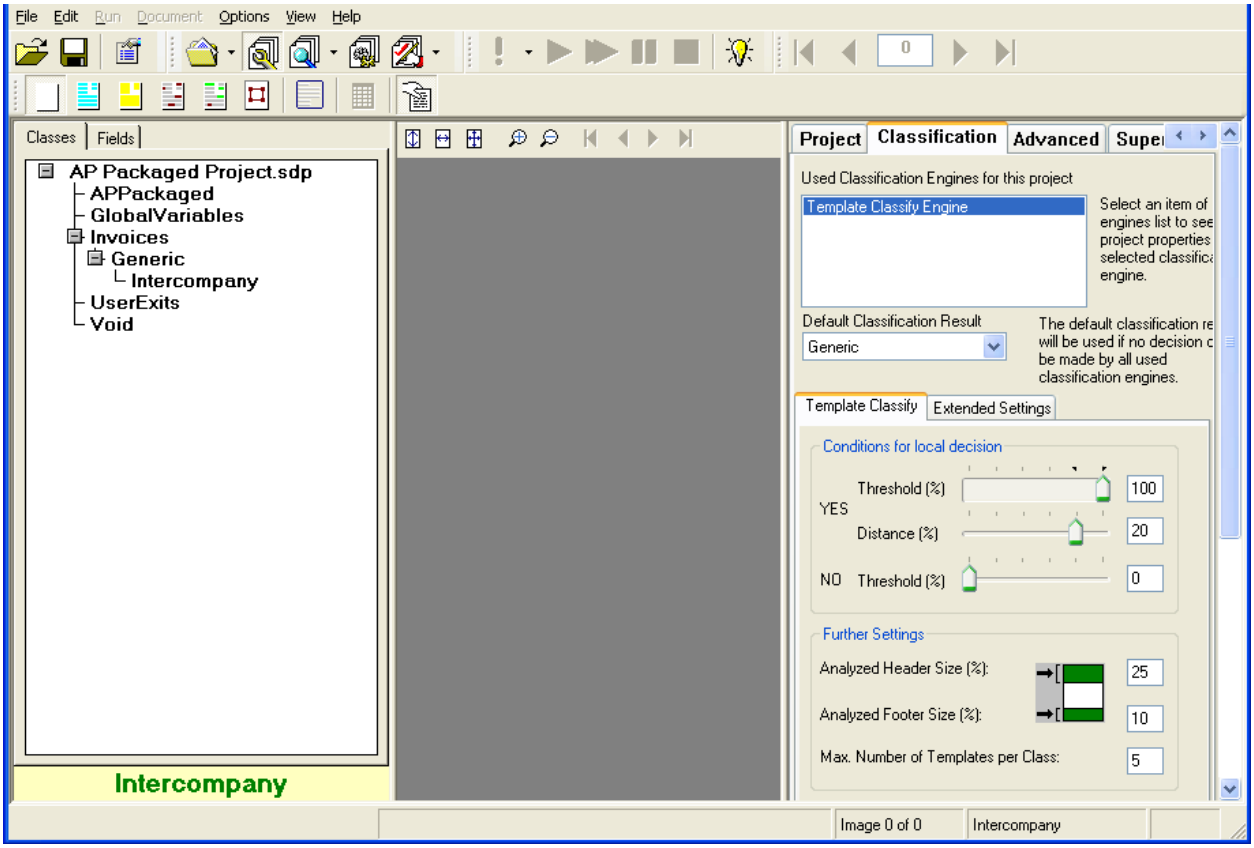


Function 'fnGetFileName' provides a simple means to extract the filename minus the filepath and file extension; the name of the intercompany class is case-sensitive and, hence, should always be referred to as 'Intercompany'.

Example 2: Client processes a significant volume of intercompany invoices, but there are only a handful of different formats

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, shown in the bottom right-hand corner in the screenshot below:



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 pre-packaged learnset held at the invoices class level.

If, however, the generic extraction is not delivering acceptable extraction results, an extraction learnset exclusively for the intercompany invoices may be created at the intercompany level using ‘Normal Train Mode’ in the solution Designer module.

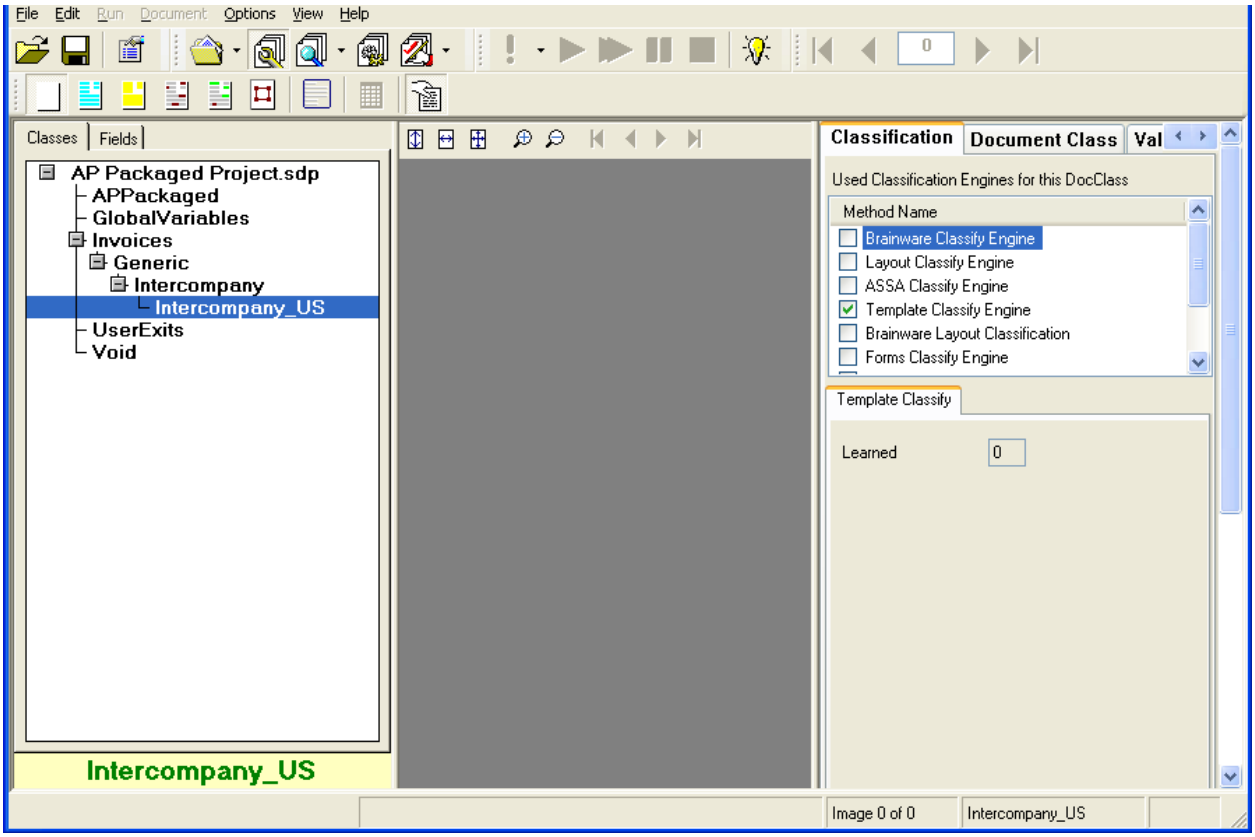
It is not possible to use the supervised learning workflow feature in conjunction with the intercompany classes; intercompany classes can only be created or amended through the Designer module.

Example 3: Client processes a significant volume of intercompany invoices, and there are a wide variety of different formats

Large, multi-national organizations with many different subsidiaries or sub-organizations located around the globe 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 sub-classes under the existing intercompany class to accommodate the different format examples required to build a classification learnset.

The screenshot below shows an example sub-class to capture intercompany invoices produced by an organization's US entity:



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 sub-classes that can be created, though it is recommended that each has a title indicative of the formats of documents it is designed to handle. For large numbers of sub-classes, it may be desirable to use the Brainware Layout Classification engine in lieu of the standard template classification engine.

APPENDIX I: CONFIGURING THE VAT REGISTRATION NUMBER COMPLIANCE CHECK

I1 Introduction

Oracle Forms Recognition provides a VAT registration number compliance check to satisfy an EU legal requirement that applies to invoices which charge 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 I4.

Both VAT registration numbers must be valid for the same country, which is why VAT is generally most prevalent on domestic invoices:

i.e. where, for example, 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 either:

1) The German company is registered for VAT in the UK, or:
2) The UK company is registered for VAT in Germany, or:
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 was 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 sets of 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 it 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, Oracle Forms Recognition will send a document to the Verifier if all of the following hold:

1) Tax is being charged on the document;
2) Both vendor and the company code being invoiced are based in an EU member state country;
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 UK pounds sterling (GBP). If the invoice is issued in the local currency of Germany (i.e. Euros/EUR), then the invoice must quote either:

1) The UK pounds sterling equivalent of the VAT amount (i.e. the local VAT amount), or:
2) The exchange rate at time of invoice issue between UK pounds sterling and Euros.

Oracle Forms Recognition 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.

I2 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 system 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 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 file/database used by the

project, and the column that denotes the VAT registration number must be mapped in the SRC section of the system configuration. For example:

```
SRC_VL_VATRegNo=Field15
```

In the example above, the VAT registration number is the fifteenth column in the vendor extract.

To activate the extraction of the bill-to VAT registration number, the company code look-up must be activated in the CCO section of the system configuration.

An alternative company code look-up BAPI may be used as long as it has the same interface as 'BAPI_COMPANYCODE_GETDETAIL'.

If the company code look-up 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 look-up 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 look-up parameters.

The following is an example populated database table containing company code look-up information:

Table: Company

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=Company
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 hold:

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 <> 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.

I3 Configuring the Validation

The final step to completing the compliance check is to add in the validation. This is done simply by putting the required configuration in place so that the system knows that the vendor and the company code belong to EU member state countries.

The system expects a flag to indicate whether or not the vendor belongs to an EU member state country to be present in the vendor extract.

In the SRC section of the system configuration, the following parameters must be completed so that the system can interpret the flag correctly (sample entries shown):

```
SRC_VL_EUMember=Field13
SRC_VL_EUMemberAlias=X
```

In the example above, the system will look for the EU member state flag in the thirteenth column of the CSV file, although, if named CSV file columns are used or a database is used as a source for the vendor data, the technical name of the column is required.

The ‘EUMemberAlias’ parameter contain the value that denotes a positive identification of an EU member state country. In the example above, if a value of ‘X’ is found in the vendor extract column configured against the ‘EUMember’ parameter, 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. Part of this configuration is covered in section K2 where, in the database look-up for the company code, the country of origin is mapped using the ‘DBCountry’ parameter in the CCO section. This must be completed in order to continue.

The missing piece is to put the configuration in the place so that the system knows that the company code country belongs to an EU member state. This is done in the CTR section of the system configuration.

This step involves installing the 'Country' database table. The SQL script which generates this table also populates it with all world countries and a flag ('X' or blank) denoting whether that country is an EU country or not.

Once this table is installed, the parameters in the CTR section should be completed as follows with the appropriate SQL connection group ('01' is used in the example above):

```
CTR_OP_ValidateFromDB=YES
CTR_VL_SQLConnectionGroup=01
CTR_VL_DBTableName=Country
CTR_VL_DBCountry=COUNTRY
CTR_VL_DBCurrency=CURRENCY
CTR_VL_DBEUMember=EUMEMBER
CTR_VL_DBName=COUNTRYNAME
```

The validation will now be activated.

The system will now set the vendor and bill-to VAT registration numbers to invalid if either:

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 (i.e. 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 which 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, then the vendor-only VAT compliance check will ONLY be carried out for invoices relating to company codes 1000, 2000 and 3000.

I4 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, all of the steps detailed in section I2 and section I3 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 hold:

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 which 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 J: ACTIVATING THE DELIVERY NOTE AND PAYMENT REFERENCE FIELDS

J1 Introduction

In the standard version of the Oracle Forms Recognition for Invoices project file, the delivery note number field and the payment reference field are not activated. This is in order to maximize system performance for projects in which these fields are not needed.

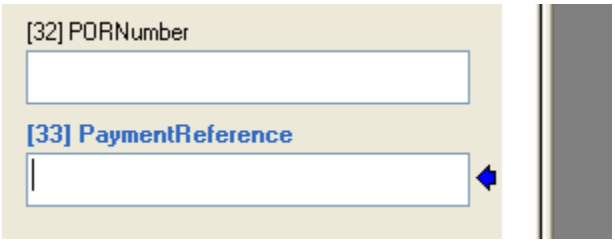
Should these fields be required in an implementation, the following steps must be followed in order to activate them.

J2 Activating the Payment Reference Field

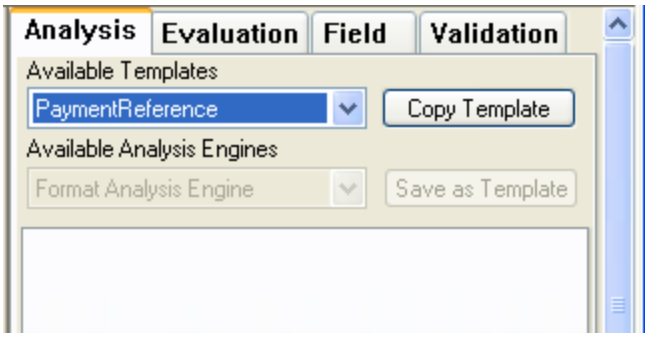
Open the project file using Designer, and go to the Definition Mode.

Now, double-click on the 'Invoices' class to show the fields.

Scroll down to the 'PaymentReference' field, which is number 33 in the list, and show the field properties.



From the drop-down menu of available analysis templates, select the predefined template 'PaymentReference'. This template does not need to be copied down in order to function.



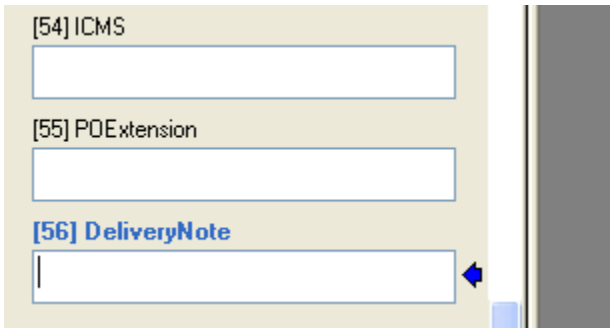
Now save the project.

J3 Activating the Delivery Note Number Field

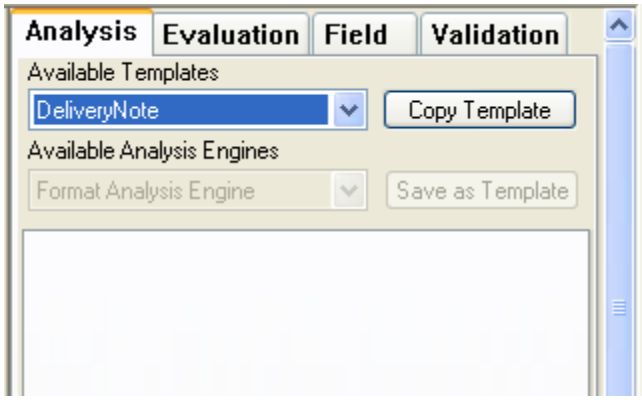
Open the project files using Designer, and go to the Definition Mode.

Now, double-click on the 'Invoices' class to show the fields.

Scroll down to the ‘DeliveryNote field, which is number 56 in the list, and show the field properties.



From the drop-down menu of available analysis templates, select the predefined template ‘DeliveryNote’. This template does not need to be copied down in order to function.



Now save the project.

APPENDIX K: UNIT OF MEASURE CONVERSIONS

K1 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 invoices include 'each', 'kilograms', 'lbs', 'boxes', 'cases' and, for service invoices that are billing time expended, 'hours' and 'days'.

Oracle Forms Recognition 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 lifted from the purchase order line. This is typically mandated by the downstream ERP system. Hence, if the invoice and purchase order units 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; the corresponding purchase order line item is for 1 tonne. Hence, when exporting the line item, a unit of measure of 'tonne' will always be used, but exporting the quantity as read from the invoice (i.e. 1000) would clearly be incorrect as 1000 tonnes was not what the vendor was billing.

During the 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 system 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 US dollars 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 (as in the example above), converting the invoice quantity from LBS to bags is not possible to do with 100% 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, Oracle provides a unit of measure conversion table, which if populated correctly, would allow the example above to succeed.

K2 Working With the Unit of Measure Conversion Table

The Unit of Measure Conversion Table holds the conversion ratio between the base unit of measure and a custom unit of measure for a given material.

The structure of the table (UOM) is as follows:

ColumnName	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.External unit of measure
Numerator	Integer	Conversion ratio numerator

Denominator	Integer	Conversion ratio denominator
UOM	String	External unit of measure

In example 2 above, there are 15 LBS contained within each bag of concrete. Hence, the table should be populated as follows:

Material	BaseUOM	Numerator	Denominator	UOM
1234A	BAG	1	15	LBS

i.e. 1 BAG = 15 LBS

During the conversion, the system will perform a look-up 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:

Converted Quantity = Invoice Quantity * (Numerator / Denominator) = 30 * (1 / 15) = 2

Hence, 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.

To activate quantity conversions from the database table, the parameters below should be set as follows:

```
LPR_OP_ConvertQuantityFromDB=YES
```

The database details themselves are configured in the UOM section of the system configuration. The following shows the delivered configuration, but the database connection string and table/column names may be changed if required:

```
UOM_VL_SQLConnectionGroup=01
UOM_VL_DBTableName=UOM

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/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 Oracle Forms Recognition log file.

K3 Unit of Measure Triangulation

A Unit of Measure Triangulation occurs when the order unit of measure on the purchase order line item is not the same as the base unit of measure for the material. Hence, three different units of measure are in play: the invoice line item unit of measure, the purchase order line item unit of measure and the base unit of measure for the material.

Example 3:

The construction company in example 2 decides to order more concrete, though 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	$1398.00	$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	$4248.00

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 (in this case, 'BAG') and the purchase order unit of measure (in this case, 'PAL').

Each single pallet holds 20 bags; hence the additional entry should be as follows:

Material	BaseUOM	Numerator	Denominator	UOM
1234A	BAG	1	15	LBS
1234A	BAG	20	1	PAL

i.e. 20 BAGS = 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:

Step 1: Convert to base unit of measure:

Base UOM Quantity = Invoice Quantity * (Numerator / Denominator) = 900 * (1 / 15) = 60

Step 2: Convert to purchase order unit of measure:

Converted Quantity = Base UOM Quantity * (Denominator / Numerator) = 60 * (1 / 20) = 3

Line pairing will now succeed for the invoice line item, and 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 in the table be made as follows:

Material	BaseUOM	Numerator	Denominator	UOM
1234A	PAL	1	300	LBS

i.e. 1 PAL = 300 LBS

Whereas this may be correct mathematically, 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 should be consistent for all entries in the table relating to a given material.

K4 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 system configuration, groups can be set up which may the alias as it appears on the invoice to the client's standard ERP system code.

Sample 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.

The aliases are not case-sensitive.

APPENDIX L: CONFIGURING PURCHASE ORDER NUMBER VALIDATIONS

L1 Introduction

Oracle Forms Recognition 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 (Oracle E-Business Suite)
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.

L2 Working with Standard Purchase Order Numbers

Standard purchase orders are those which 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.

An example purchase order header table structure (with sample data) reflecting this arrangement would be as follows:

Table: PO_HEADER

PONUMBER*	VENDORID	COMPANYCODE	POTYPE	CURRENCY
1928370	100010	GB01	GD	GBP

When a purchase order number is extracted in the purchase order 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 client) 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. In Verifier, if the PO vendor differs from the invoice vendor for both options above, the 'PO VENDOR <> 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 Oracle Forms Recognition purchase order 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 look-up.

Example:

A client uses purchase order type 'GD' for goods, and 'SV' for services. To configure the system to 'flip' the PO type 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 purchase order type will switch 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 read 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 L6.

L3 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 (F4311). These keys are:

1) The purchase order number
2) The company code
3) The purchase order type
4) The purchase order suffix

Hence, the purchase order header table will have a structure similar to the below:

Table: PO_HEADER

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. As of build 1006, Oracle Forms recognition 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 L2 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 as '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.

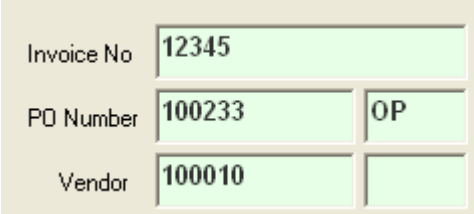
Tip: A common OCR problem is that the 'O' in 'OP' will be read as a zero, hence configuring a second format (10####0P) can be beneficial. The system will auto-correct this to 'OP'.

'OP' should also be added into the comma-separated list of valid JD Edwards purchase order types:

```
PON_VL_JDEPOTypes=OP
```

When 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 as shown in the screenshot below:



In instances where the purchase order type is always going to be 'OP' or another constant (which will depend on the client), this JD Edwards configuration approach is not appropriate. An alternative approach that just uses the purchase order number and the company code to perform the look-up into the purchase order header table is described in section L5.

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 system configuration), or to use the associative search engine to derive it based on the bill-to details on the invoice. Oracle Forms Recognition always 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 '100233' is read from the invoice, the system will format the purchase order number to '0000100233'.

If the parameter is left blank, no leading zeros will be added.

All other aspects of the validation (for example, validating the PO vendor against the invoice, setting the currency and Oracle Forms Recognition PO type fields) operate in the exact same way as the standard purchase order number validation.

L4 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 (PS_PO_HDR). This key consists of the purchase order number and the purchasing business unit.

A sample purchase order header table is shown below:

Table: PS_PO_HDR

PO_ID*	BUSINESS_UNIT*	VENDOR_ID	VNDR_LOC	PO_TYPE	CURRENCY_CD

1002334	ORACLE	100010	1000	GD	GBP

PeopleSoft uses a vendor ID/site ID combination (VENDOR_ID/VNDR_LOC) 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. Hence, the strategy to derive this 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 system 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 parameter below must be set to 'YES':

```
PON_OP_PeoplesoftPO=YES
```

At a minimum, the purchase order number, vendor ID and purchasing business unit fields must 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 message if the mapping is insufficient or incorrect.

On the invoice itself, the system expects the vendor to state the purchase order as '1002334 ORACLE', 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:

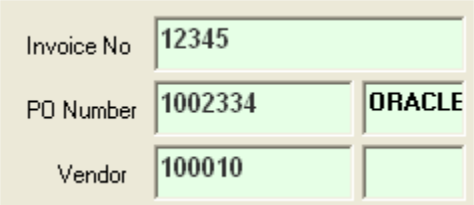
```
PON_VL_01_Format=10#####ORACLE
PON_VL_01_Ignore=.,
```

This means that the system will be looking for a seven digit number beginning with '10', which has 'ORACLE' at the end.

'ORACLE' should also be added into the comma-separated list of valid PeopleSoft purchase order types:

```
PON_VL_PeopleSoftBusinessUnits=ORACLE
```

When purchase order number is extracted, the system will automatically separate the purchase order number itself from the purchasing business unit, both on server side, as well as in Verifier if the whole value is entered into the purchase order 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 as shown in the screenshot below:



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 ‘1002334’ is read 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 (for example, validating the PO vendor against the invoice, setting the currency and PO type fields) operate in the exact same way as the standard purchase order number validation.

L5 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:

Table: PO_HEADER

PONUMBER*	COMPANYCODE*	VENDORID	POTYPE	CURRENCY
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 Oracle Forms Recognition 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 message if any of the above is missing or incorrect.

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 this reason, it is recommended that it is determined either at the point of scan and included in the document filename, or determined using the associative search engine via the invoice bill-to details.

All other aspects of the validation (for example, validating the PO vendor against the invoice, setting the currency and PO type fields) operate in the exact same way as the standard purchase order number validation.

L6 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' (i.e. 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:

Table: PO_LINES

PONUMBER*	LINE*	LINETYPE	MATERIAL	DESCRIPTION	ORDER_ QTY	ORD_ UOM	UNIT_ PRICE	PRICEUNIT	ORDER_ AMT
1928370	10	GOODS	1234A	Concrete	2000	KG	500.00	1000	1000.00

Note that this table has two key fields which 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, which are the purchasing business unit and the company code/purchase order type respectively

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 must be mapped:

```
LPR_VL_DBPO=PONUMBER
LPR_VL_DBLINE=LINE
```

For JD Edwards purchase orders, the following columns must 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 which represents the client's unique material number for that item.

The price unit column should be mapped if available or else 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, which, in turn, can influence the setting of the purchase order type field.

For example, if 'SRV' or 'DSRV' denotes that the purchase order line item is a service, then the following parameter will switch the PO type field to 'SERVICE' if any of the PO lines meet this criteria:

```
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 PO 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 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:

<table>
<tr><th>Column</th><th>Explanation of usage</th></tr>
<tr><td>DBMATERIALGROUP</td><td>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 selection of the correct tax code is driven by the material group of the item.</td></tr>
<tr><td>DBTAXCODE
DBTAXJURCODE</td><td>These two columns represent the tax information that was set when the purchase order was originally raised.

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 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 (column DBUOM) which is associated with the order quantity on the purchase order line.

Example:

The client 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 <u>MUST</u> always be mapped in implementations where instances such as the above are possible within the client'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</td><td>These four columns provide Oracle Forms Recognition with purchase order line item history data, so that the system knows exactly what has been invoiced and goods-receipted to date,</td></tr>
</table>

DBTOTALQUANTITYINVOICED
DBTOTALVALUEINVOICED

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:

Line	Description	Order Qty	Unit Price	Total
1	Product A	2	$10.00	$20.00
2	Product A	2	$10.00	$20.00

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 'LPR_OP_EnableIntegrityCheck' is set to 'YES', else 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:

Line	Description	Order Qty	Unit Price	Total	TQD	TVD	TQI	TVI
1	Product A	2	$10	$20	2	$20	0	0
2	Product A	2	$10	$20	0	0	0	0

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:

Line	Description	Order Qty	Unit Price	Total	TQD	TVD	TQI	TVI
1	Product A	2	$10	$20	2	$20	2	$20
2	Product A	2	$10	$20	0	0	0	0

In this instance, the system will be able to see that the first purchase order line is already fully invoiced; hence the invoice line will be paired to available PO line item 2.

In example 1 above, the MIRA condition would hold as the net value of the invoice would be equal to the total value of goods receipts not yet invoiced. Hence, 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 controlled in the TAB section of the system configuration. Example 2 would also meet the MIRA condition if there was a $20 goods receipt value against line 2, but no invoices booked against it. If the 'LPR_IgnoreCompletedPOLines' flag is set to 'YES', then 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.
DBPLANT	The plant is a code set in the client 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: 1) The invoice includes a miscellaneous charge, and miscellaneous charges are set to be outputted as general ledger account entries, where the corresponding coding string in table MISC is driven by the plant on the purchase order; 2) 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 database table PLANT. Settings in the TAX section of the system configuration control the appropriate data source.
DBCHARGECODE DBCHARGECODEID	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 brought into Oracle Forms Recognition and then passed back out for each line item where line pairing has succeeded.

L7 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 client 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 system configuration. The parameters can be incoming or outgoing, and can be set either to pass the content of a specific Oracle Forms Recognition 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 Oracle Forms Recognition field, is case sensitive and must be the technical name of the field (PONumber, CompanyCode, DocumentType, AmountTotal, etc.). 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 Oracle Forms Recognition) should have a direction of 'I'; 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
```

Once the parameters have been defined, they can be assigned to the corresponding stored procedure. Assignment of parameters to the purchase order header stored procedure is controlled via the following configuration setting:

```
PON_VL_StoredProcedureParameters=01,02,03
```

In the example above, parameters '01', '02' and '03' have been assigned. Parameters are assigned to the purchase order line item stored procedure via the following configuration setting:

```
LPR_VL_StoredProcedureParameters=01
```

The system will raise an error if no parameters have been assigned, or if a parameter listed has not been created in the SPC section of the system configuration.