Perceptive Accounts Payable Invoice eForm

Connector Configuration Guide

Version: 12.11.x

Written by: Documentation Team
Date: August 2020

Hyland

Copyright

Information in this document is subject to change without notice. The software described in this document is furnished only under a separate license agreement and may be used or copied only according to the terms of such agreement. It is against the law to copy the software except as specifically allowed in the license agreement. This document or accompanying materials contains certain information which is confidential information of Hyland Software, Inc. and its affiliates, and which is subject to the confidentiality provisions agreed to by you.

All data, names, and formats used in this document's examples are fictitious unless noted otherwise. Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright law, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Hyland Software, Inc. or one of its affiliates.

Hyland® and Hyland Software®, as well as Hyland product names, are registered and/or unregistered trademarks of Hyland Software, Inc. and its affiliates in the United States and other countries. All other trademarks, service marks, trade names and products of other companies are the property of their respective owners.

© 2020 Hyland Software, Inc. and its affiliates. All rights reserved.

Table of Contents

What are connectors with respect to Perceptive Accounts Payable (AP) Invoice eForm? 6
Virtual Table Connector 6
Determine your virtual table requirements 6
About using differential data 7
Configure the virtual tables 7
Create and configure the CSV files 7
Configure the upload process 9
Import and archive the data 11
Purge obsolete data 12
SAP Connector 14
Integrate AP Invoice eForm with Perceptive SAP Connector 14
Microsoft Dynamics AX Connector 15
Integrate AP Invoice eForm with Microsoft Dynamics AX Connector 15
Version 3.0.x 15
Version 4.0.x 15
Microsoft Dynamics 365 Connector 15
Integrate AP Invoice eForm with Microsoft Dynamics 365 Connector 15
Custom Connector 16
Create a Custom Connector 16
Integrate AP Invoice eForm with Custom Connector 16
Demo Connector 16
Integrate AP Invoice eForm with Demo Connector 16
Edit Envoy service 16
Edit Connector config 17
Appendix A: AP Invoice eForm virtual tables 18
Account Assignment Lines 18
AOC GL Account 18
AOC GL Code 20
Business Unit – *required* 21
Currency 22
Currency Country 22
GL Account - *required* 23
GL Code - *required* 24

GL User 25
Location - *required* 25
Payment Terms 26
Purchase Order - *required* 27
Purchase Order Line - *required* 27
Sales Use Tax (SUT) Codes 29
Sales Use Tax (SUT) Apply Codes 29
Special Handling 30
VAT Code 30
VAT Registration Number (VAT ID) Vendor 31
VAT Registration Number (VAT ID) Business Unit 31
Vendor - *required* 32
Vendor Location 33
Appendix B: Reserved characters in XML and forms 34
Appendix C: Functions and associated parameters of CustomConnector.js 35
Customize functions and associated parameters 35
initialize *35*
finalize *35*
getBusinessUnits *35*
getVATCodes *36*
getLocations *37*
getSUTCodes *37*
getSUTApplyCodes *38*
getCurrencyCodes *38*
getSpecHandlingCodes *39*
getPaymentTermsCodes *39*
getCountryCurrencyRels *40*
getAOCCodes *40*
getCompanyDetails *41*
getTaxCodes *41*
getVendors *42*
getPO *43*
getVendorLocations *44*
getPODetails *45*
getGLAcctCodes *47*

getGLCodeList *48*
getGLCodeUserConstraints *48*
getPOCreditData *49*
getVendorVATIDData *49*
getSoldToVATIDData *50*
getGLAcctDescription *51*
getHeaderLookupDataSet *52*

What are connectors with respect to Perceptive Accounts Payable (AP) Invoice eForm?

Connectors in AP Invoice eForm are components that integrate AP Invoice eForm with Enterprise Resource Planning (ERP) applications and other databases. AP invoice eForm includes dedicated connectors to communicate with the following ERP application and databases.

- Virtual Table Connector
- SAP Connector
- Microsoft Dynamics AX Connector
- Microsoft Dynamics 365 Connector
- Custom Connector
- Demo Connector

Virtual Table Connector

AP Invoice eForm integrates the data from your ERP application with Perceptive Content. The eForm uses this data to populate value look-ups and lists for your invoice-processing users. It also uses the data values for validation procedures.

Perceptive Content internally stores the extracted ERP data using virtual tables. In ImageNow 6.6.x and earlier, virtual tables are a collection of projects. In ImageNow 6.7.x and higher, virtual tables are a collection of folders. Each project or folder type represents a table. Each instance of the project or folder represents a record, or row, in that table. Update the virtual tables as often as necessary to keep the data synchronized with your ERP application.

This guide defines the fixed structure and format of the virtual tables. The ERP system administrator provides the source comma separated value (CSV) files in the defined format. This guide then describes how to load the data as virtual tables and set up a scheduled process for loading and synchronizing the data.

After the initial data loading, the administrator only needs to provide additions and updates. You will determine the upload frequency for your business process, which can vary by table.

To learn how to install AP Invoice eForm, refer to the *Perceptive AP Invoice eForm Installation and Setup Guide*. For additional information on configuring the eForm, refer to the *Perceptive AP Invoice eForm Advanced Design and Setup Guide*. Both guides are available in the Perceptive Software Customer Portal on the Product Documentation tab.

Determine your virtual table requirements

The tables required for your implementation depend on your solution design. Both your eForm setup and the features you use, such as value added tax (VAT), affect your requirements. To assess your virtual table requirements and prepare for configuration, complete the following general steps.

- Review the Appendix A: AP Invoice eForm virtual tables section of this document and determine which tables you require.
- Review the required fields in each of your required tables and familiarize yourself with the CSV file structure. There are sample tables provided for each table in this guide.
- Identify the ERP data source and determine how to export it to CSV format.

About using differential data

The initial import of CSV data into the virtual tables requires a full set of active records. This first set can take many hours to export from your ERP and import into Perceptive Content depending on the size of your database. We recommend scheduling the first data import during an off-peak time, such as overnight or during a weekend.

After you create the virtual tables, completely purging and reloading your data during each cycle can take extensive amounts of time. A complete reload also carries the risk of running with an incomplete data set, should a failure occur. Therefore, to protect your data during routine updates, we recommend that you import only differential data.

For example, for the Vendor table, you only need to export new vendors, inactivated vendors, or vendors with updated information from your ERP. Depending on your ERP or other data source, you can use a variety of methods to collect data that is new, updated, or inactivated since the last export.

To detect changes to a record, some AP Invoice eForm administrators add triggers directly to the database; others utilize functionality provided within the ERP.

The frequency with which you update the records varies for each specific virtual table. The Purchase Order Line table, for example, usually requires updates daily. You may only need to update the GL Account table every few days, however.

For more information on creating and scheduling data imports, consult your ERP or database management system (DBMS) documentation. For more information on indicating active versus inactive data in your tables, refer to the About setting records to active or inactive section later in this guide.

Configure the virtual tables

To configure and use the virtual tables, review and complete the following tasks as needed.

- Create and configure the CSV files
- Configure the upload process
- Import and archive the data
- Purge obsolete data

Refer to the steps and information in the following sections.

Create and configure the CSV files

To create the records that populate AP Invoice eForm fields, you must create a library of virtual tables. The AP Invoice eForm process populates each virtual table with a CSV file from your ERP application. We recommend using a separate CSV file for each virtual table you need.

File specifications

- The process you use to generate your CSV files depends on your ERP. For more information on creating your CSV files, refer to your ERP product documentation.
- You do not need to use leading zeros when you export numeric values.
- Always use a period as the decimal separator. AP Invoice eForm provides punctuation in the user interface based on your locale and currency. For more information on currency settings, refer to the “Configure AP Invoice eForm for a global environment” section in the *Perceptive AP Invoice eForm Advanced Design and Setup Guide*.

- **Important** The virtual tables limit string fields to 128 characters. If any value in the record exceeds 128 characters, the process does not import the record and logs an error to the log file.
- ImageNow 6.7 and previous versions support the ISO-8859-1 character set.

  ImageNow 6.8 or higher versions support UTF-8.
- To retain original text that contains your delimiter and avoid parsing issues:
  - Wrap the value, whether numeric, text, or a character, in double quotes.
  - You must use double quotes for every value within the virtual table row that includes the delimiter.
  - For more information on reserved characters in XML, refer to Appendix B: Reserved characters in XML and forms.

About CSV file names

We recommend that you use the TableName value for the CSV file name, such as BUS_UNIT.csv. However, there is no required naming convention.

To add a date and time stamp automatically prior to archiving CSV files, refer to the Enable time stamps on processed CSV files section.

About the virtual table column headers

You can include optional column header rows to describe each column of the table. Examples include TableName, VendorGroup, and Jurisdiction. In this guide, the column headers provided in the example tables are only suggestions. You can use any column header description for your own reference.

Important Perceptive Content identifies virtual tables with the TableName key, such as BUS_UNIT for the Business Unit table.

Example

```
TableName|Active|BusUnitId|BusUnitName|VendorGroup|VAT Jurisdiction|VAT ID
BUS_UNIT|Y|1234|ACME Construction|ACME|US|
BUS_UNIT|Y|4500|CGI Corporation|EUR|FR|FRXX999999999
```

About the delimited rows of data

Each row of data from your ERP should create one line in the CSV file. Separate each field in the row with a consistent delimiter. The default delimiter is the pipe character, "**|**". We recommend using the pipe character because ERPs rarely use it in text values.

If you must use a common character as the delimiter, such as a comma, wrap every value in the record with double quotes. This allows you to avoid parsing issues if the delimiter appears in the value, such as in postal addresses.

In this example, the second record includes a comma inside the Address1 column.

```
TableName,Active,VendorGroup,VendorID,RemitTo,VendorName,Address1,Addres2,City,State
"VENDOR","Y","ACME","110000","2","General Supply Co","42 Douglas Dr",,"San Diego","CA"
"VENDOR","Y","ACME","110001","1","General Supply Co","1600 Main St, B",,"Chicago","IL"
```

We recommend that you use a text editor, rather than Microsoft Excel, to view the data during file preparation.

About setting records to active or inactive

Most of the AP Invoice eForm virtual tables include an Active column. The only tables that do not include this column are the Purchase Order and Purchase Order Line tables. The Active value indicates whether the data in that row is accessible to the eForm.

To indicate active records, you can use the values Y, A, 1, or TRUE. Any other value inactivates the record.

You can also use the Active column to search for and purge inactive records from the table. For more information on scheduling this as an automated action, refer to the Purge obsolete data section.

For the initial data load, every record should be active.

Configure the upload process

To configure the CSV upload process, complete the following tasks.

- Set the CSV import, archive, and error processing paths
- Enable the virtual tables and columns
- Set the delimiter and the logging

Review and complete the following tasks.

Set the CSV import, archive, and error processing paths

To map the import, archive, and error paths, complete the following steps.

1. On the Perceptive Content Server computer, navigate to **[*drive*:]\inserver\etc\ap**.
2. Open **AP_VirtualTable_Updater.xml** with a text editor.
3. In the **<configuration>** node, in the `inputpath` attribute, verify or set the CSV import directory. For example, `C:\inserver\temp\ap\import\*.csv`.
4. In the `<configuration>` node, in the `archivepath` attribute, verify or set the CSV archive directory.

   Note This is the directory where the eForm scripts move the CSV files after processing.
5. In the `<configuration>` node, in the `errorpath` attribute, verify or set the CSV error directory.

   Note This is the directory where the eForm scripts move CSV files if the upload script encounters errors during processing.
6. Save and close **AP_VirtualTable_Updater.xml**.
7. On the Perceptive Content Server computer, place your CSV files in the import directory.

Enable the virtual tables and columns

The AP_VirtualTable_Updater.xml file includes a record structure for each of the virtual tables.

If needed, to review which virtual tables and columns you need, refer to the Appendix A: AP Invoice eForm virtual tables section.

In the Appendix, each example table notes required columns with an asterisk. The import process maps the CSV records to virtual tables according to the TableName column value. You cannot add new record structures or columns within a record structure. However, you can change the order of the record structures and the order of the existing columns.

You enable each virtual table and column that your solution requires in the AP_VirtualTable_Updater.xml file. Complete the following steps.

1. Navigate to **[*drive*:]\inserver\etc\ap** and open **AP_VirtualTable_Updater.xml** with a text editor.
2. For each virtual table you need to enable, in the `<recordstructure>` element, set the `enabled` attribute to `"true"`.
3. For each column you use with that virtual table, in the `<column>` element, set the `enabled` attribute to `"true"`.
4. For each `<column>` element that has a `required` attribute, set the `required` attribute to `"false"` to allow uploading of blank values to a virtual table. If you set the `required` attribute to `"true"`, the **AP_VirtualTable_Updater.js** file restricts uploading of blank values to a virtual table.
5. Optional. To configure the date format, in the `<column>` element, set the `format` attribute with a string using `MM` for month, `DD` for day, and `YY` or `YYYY` for year. Each date segment can be separated by either a slash "**/**", a dash "**-**", or no separator.

   If you do not specify a format, the CSV must provide the date in the format YYYY-MM-DD or YYYY-MM-DD hh:mm:ss.

   If you specify a date format, you cannot specify or provide a time format.

   Find common examples listed below.

   - `"MMDDYYYY"` or `"MM/DD/YYYY"` or `"MM-DD-YYYY"`
   - `"MMDDYY"` or `"MM/DD/YY"` or `"MM-DD-YY"`
   - `"DDMMYYYY"` or `"DD/MM/YYYY"` or `"DD-MM-YYYY"`
   - `"DDMMYY"` or `"DD/MM/YY"` or `"DD-MM-YY"`
   - `"YYYYDDMM"` or `"YYYY/DD/MM"` or `"YYYY-DD-MM"`
   - `"YYDDMM"` or `"YY/DD/MM"` or `"YY-DD-MM"`
   - `"YYYYMMDD"` or `"YYYY/MM/DD"` or `"YYYY-MM-DD"`
   - `"YYMMDD"` or `"YY/MM/DD"` or `"YY-MM-DD"`

Set the delimiter and the logging

In the AP_Config.xml file, you can set the field delimiter and configure logging for the updater script.

The default field delimiter is the pipe character, "**|**". You can change the default to another character, such as a comma or semi-colon.

To configure the field delimiter, complete the following steps.

1. Navigate to **[*drive*:]\inserver\etc\ap** and open **AP_Config.xml** with a text editor.
2. Under the **`<AP_VirtualTable_Updater>`** node, set the **`<DataFieldDelimiter>`** with a character, such as a comma "**,**".
3. Under the **`<AP_VirtualTableUpdater>`** node, complete the following substeps.
   - To log to a file, set the **`<LogToFile>`** element to `true`.
   - To log to the console, set the **`<LogToFile>`** element to `false`.
   - Set the **`<DebugLevel>`** element with a number between `0` and `5`, where `5` is the most verbose.

Example

```
<AP_VirtualTable_Updater>
    <LogToFile type="boolean">true</LogToFile>
    <DebugLevel type="int">5</DebugLevel>
    <UpdaterConfigFile>AP_VirtualTable_Updater.xml</UpdaterConfigFile>
    <DataFieldDelimiter>|</DataFieldDelimiter>
    <Connector>CSV</Connector>
    <AppendDateTimeStampToFile>false</AppendDateTimeStampToFile>
</AP_VirtualTable_Updater>
```

4. Save and close **AP_Config.xml**.

Import and archive the data

To populate the virtual tables, import the data in the CSV files. The upload script routes imported data into an archive directory. Review or complete the following tasks.

- Import the CSV files
- Enable time stamps on processed CSV files

Import the CSV files

The AP_VirtualTable_Updater.js script monitors the import directory for CSV files. It imports new records or updates records in the virtual tables. Administrators often schedule this script to run each night after business hours. It can be run more frequently if needed. Depending on the size of your CSV files, this update may take several hours to process.

To run the script, complete the following steps.

1. To start **INTool**, on the Perceptive Content Server computer, access a command prompt and change to the **[*drive:*]\inserver\bin** directory.

   Note If you use a 64-bit operating system, access INTool at **[*drive:*]\inserver\bin64**.

2. To run the virtual table updater script, enter the following command:

   ```
   intool --cmd run-iscript --file AP_VirtualTable_Updater.js
   ```

3. After the command completes, to verify the script succeeded, complete the following substeps.

   1. Navigate to **[*drive:*]\inserver\log** and open **AP_VirtualTable_Updater_[*date*].log** in a text editor.

      Note In an active-active server environment, navigate to the **[*drive*:]\inserver\log** directory on your primary node.

   2. To verify the script succeeded, look for lines similar to this example at the end of the file.

      ```
      06/13 08:55:54.729 AP_VirtualTable_Updater:0B [ NOTIFY ] Done:

      ************************************************
          2836     C:\inserver\temp\ap\import\PO Lines.csv - ADD
             2     C:\inserver\temp\ap\import\PO Lines.csv - CHANGE
          2838     C:\inserver\temp\ap\import\PO Lines.csv - TOTAL
          2836     TOTAL - ADD
             2     TOTAL - CHANGE
          2838     TOTAL RECORDS

      Script run time 103 Seconds
      ************************************************
      ```

```
06/13 08:55:54.729 AP_VirtualTable_Updater:0B [  INFO  ]
06/13 08:55:54.729 AP_VirtualTable_Updater:0B [  INFO  ] Script run time:  1m
43s (103 total seconds)
```

4. Create a batch file that runs the script using INTool and schedule using Windows Task Scheduler to run on a regular basis. Below is an example batch file (AP_VirutalTable_Updater.bat).

```
c:
cd c:\inserver\bin
intool --cmd run-iscript --file AP_VirtualTable_Updater.js
```

Enable time stamps on processed CSV files

Optional. The AP_VirtualTable_Updater.js script moves files processed successfully to the archive, and unsuccessful files to the error directory. In the AP_VirtualTable_Updater.xml file, you specify the archive and error directories. By default, if there is a file with the same name in the directory, the script cannot overwrite the existing file to archive the new one. In this instance, the processed file stays in the import directory.

However, you can set the updater script to append processed file names with a time stamp. This results in a unique name, such as:

<*file name*>_YYYYMMDD_hhmmss.csv

To enable the time stamp, complete the following steps.

1. Navigate to **[*drive*:]\inserver\etc\ap** and open **AP_Config.xml** with a text editor.
2. Under the `<AP_VirtualTable_Updater>` node, set the `<AppendDateTimeStampToFile>` element to `true`.
3. Save and close the **AP_Config.xml** file.

Purge obsolete data

The AP_VirtualTable_Purge.js script deletes obsolete records from the virtual tables. It deletes entries based on purge criteria you configure for each table. To configure the purge script, review or complete the following tasks.

- Configure the virtual table purge
- Schedule the virtual table purge

Configure the virtual table purge

There are three ways to specify virtual table purge criteria in AP_VirtualTable_Purge.js.

- **Number of days**. You can purge records a specified number of days after an event occurs. For example, you can purge a table 30 days after its last update. You can use this with purchase orders and purchase order lines.
- **Inactivity**. You can monitor specific fields for activity and purge inactive records. You can use this with business units, vendors, payment terms, general ledger (GL) accounts, and GL users.
- **Number of days plus inactivity**. You can monitor fields for activity, then purge the table for a specified number of days after the field becomes inactive.

To configure the purge script criteria, complete the following steps.

1. Navigate to **[*drive*:]\inserver\script** and open **AP_VirtualTable_Purge.js** with a text editor.
2. For each virtual table you want to purge, as indicated in the **`name`**, use one or both of the following methods.

   Note The `name` is the name of the virtual table project in ImageNow 6.6 and previous versions, or folder in ImageNow 6.7 and higher.

   - To purge a table, according to the number of days that have passed since an event, complete the following substeps.
     1. Set **`date_column`** with the name of the custom property that holds the date value you want to monitor. For example,

        ```
        date_column: "Z_APW Date Last Updated"
        ```

     2. Set **`numberofdays`** with an integer representing whole days. For example,

        ```
        numberofdays: "30"
        ```

   - To purge according to a flag-type custom property switching from “active” to “inactive,” complete the following substeps.
     1. Set **`active_column`** with the name of the flag-type custom property that you want to monitor. For example,

        ```
        active_column: "Z_APW Active"
        ```

     2. Set **`active`** to `"false"`.
3. Save and close **AP_VirtualTable_Pruge.js**.

Example AP_VirtualTable_Purge.js configurations

Delete inactive vendor records.

```
{
name: "Z_APW_APVendor",
active_column: "Z_APW Active",
 active: "false",
date_column: "",
 numberofdays: ""
}
```

Delete purchase orders that have been closed for 30 days.

```
{
name: "Z_APW_APPO",
active_column: "",
 active: "",
date_column: "Z_APW PO Date Closed",
 numberofdays: "30"
}
```

Delete inactive GL account records that the system has not updated for 60 days.

```
{
name: "Z_APW_APGLAccount",
active_column: "Z_APW Active",
 active: "false",
date_column: "Z_APW Date Last Updated",
 numberofdays: "60"
}
```

Schedule the virtual table purge

When used, you typically run the AP_VirtualTable_Purge.js script daily after business hours. Depending on the size of your virtual tables, this update may take several hours to process.

Complete the following steps to run the AP_VirtualTable_Purge.js.

1. To start **INTool**, on the Perceptive Content Server computer, access a command prompt and change to the **[*drive:*]\inserver\bin** directory.

   Note If you use a 64-bit operating system, access INTool at **[*drive:*]\inserver\bin64**.

2. To run the virtual table purge script, enter this command:

   ```
   intool --cmd run-iscript --file AP_VirtualTable_Purge.js
   ```

3. After the command completes, to verify the script succeeded, complete the following substeps.

   1. Navigate to **[*drive:*]\inserver\log** and open **AP_VirtualTable_Purge_[*date*].log** in a text editor.

      Note In an active-active server environment, navigate to the **[*drive*:]\inserver\log** directory on your primary node.

   2. To verify the script succeeded, look for lines similar to this example at the end of the file.

      ```
      06/13 08:55:54.729 AP_VirtualTable_Purge:0C [ NOTIFY ] Done:

      ************************************************
            9707          Z_APW_APVendor
            9707    RECORDS PURGED - TOTAL

      Script run time 103 Seconds
      ************************************************

      06/13 08:55:54.729 AP_VirtualTable_Purge:0C [  INFO  ]
      06/13 08:55:54.729 AP_VirtualTable_Purge:0C [  INFO  ] Script run time:  1m 43s
      (103 total seconds)
      ```

4. Optional. Create a batch file that runs the script using INTool, and schedule it to run daily using Windows Task Scheduler. A sample of batch file content:

   ```
   c:
   cd c:\inserver\bin
   intool --cmd run-iscript --file AP_VirtualTable_Purge.js
   ```

SAP Connector

Using the Perceptive Connector for SAP Financials, Perceptive AP Invoice eForm connects to your SAP application. The connector enables AP Invoice eForm to perform live lookups and create, park, and post invoices in SAP. To integrate AP Invoice eForm with SAP using the Perceptive SAP Connector, you must configure the connector settings in the AP_Config.xml file.

Integrate AP Invoice eForm with Perceptive SAP Connector

To integrate AP Invoice eForm with Perceptive SAP Connector, complete the following steps.

1. In the **AP_Config.xml** file, under `<configurations>`, locate the `<connector>` tag.
2. Within the `<connector>` tag, replace the existing value with `SAPConnector`.

For additional information, refer to the *Perceptive Connector for SAP Financials Installation Guide*.

Microsoft Dynamics AX Connector

Using the Perceptive Connector for Microsoft Dynamics AX, Perceptive AP Invoice eForm connects to your Microsoft Dynamics AX application. The connector enables AP Invoice eForm to perform live lookups and create invoices in Microsoft Dynamics AX. To integrate AP Invoice eForm with Microsoft Dynamics AX using the Perceptive Connector for Microsoft Dynamics AX, you must configure the connector settings in the AP_Config.xml file.

Integrate AP Invoice eForm with Microsoft Dynamics AX Connector

Version 3.0.x

To integrate AP Invoice eForm with Perceptive Connector for Microsoft Dynamics AX 3.0.x, complete the following steps.

1. In the **AP_Config.xml** file, under `<configurations>`, locate the `<connector>` tag.
2. Within the `<connector>` tag, replace the existing value with `DAXConnector`.

For additional information, refer to the *Perceptive Connector for Microsoft Dynamics AX Installation Guide, version 3.0.x*.

Version 4.0.x

To integrate AP Invoice eForm with Perceptive Connector for Microsoft Dynamics AX 4.0.x, complete the following steps.

1. In the **AP_Config.xml** file, under `<configurations>`, locate the `<connector>` tag.
2. Within the `<connector>` tag, replace the existing value with `DAXConnector_v4`.

For additional information, refer to the *Perceptive Connector for Microsoft Dynamics AX Installation Guide, version 4.0.x*.

Microsoft Dynamics 365 Connector

Using the Perceptive Connector for Microsoft Dynamics 365, Perceptive AP Invoice eForm connects to your Microsoft Dynamics 365 application. The connector enables AP Invoice eForm to perform live lookups and create invoices in Microsoft Dynamics 365. To integrate AP Invoice eForm with Microsoft Dynamics 365 using the Perceptive Connector for Microsoft Dynamics 365, you must configure the connector settings in the AP_Config.xml file.

Integrate AP Invoice eForm with Microsoft Dynamics 365 Connector

To integrate AP Invoice eForm with Perceptive Connector for Microsoft D365, complete the following steps.

1. In the **AP_Config.xml** file, under `<configurations>`, locate the `<connector>` tag.
2. Within the `<connector>` tag, replace the existing value with `D365Connector`.

For additional information, refer to the *Perceptive Connector for Microsoft Dynamics 365 for Finance and Operations Installation Guide, version 1.0.x*.

Custom Connector

The Custom Connector enables you to create a connector customized to integrate with external data sources using ODBC, JDBC, or web services.

Create a Custom Connector

To create your custom connector, complete the following steps.

1. Navigate to **[*drive*:]/inserver/script/apef/connectors**.
2. Open **CustomConnector.js**.
3. Modify the functions to create a connector customized for your environment. For additional information about the functions you can use, refer to Appendix C: Functions and associated parameters of CustomConnector.js

Integrate AP Invoice eForm with Custom Connector

To integrate AP Invoice eForm with the custom connector, complete the following steps.

1. In the **AP_Config.xml** file, under `<configurations>`, locate the `<connector>` tag.
2. Within the `<connector>` tag, replace the existing value with `CustomConnector`.

Demo Connector

The Demo connector allows you to connect your AP Invoice eForm to a set of scripted data for demonstration purposes. The Demo connector is the default connector upon initial installation and is helpful for diagnostic and troubleshooting purposes.

Integrate AP Invoice eForm with Demo Connector

To integrate AP Invoice eForm with the demo connector, complete the following steps.

1. In the **AP_Config.xml** file, under `<configurations>`, locate the `<connector>` tag.
2. Within the `<connector>` tag, replace the existing value with `DemoConnector`.

Edit Envoy service

If you configure AP Invoice eForm to work with Perceptive Connector for SAP Financials or Perceptive Connector for Dynamics AX 3.0.x, you may require to edit the wsdlUrl or the name in Envoy service. To edit Envoy service, complete the following steps.

1. Navigate to **[*drive*:]\<*INServer*>\temp\apef_install\etc**.
2. Open the **AP_Config.xml** file.
3. Under the `<envoy>` element, edit the values for `<name>` and `<wsdlUrl>`.

   Note When you edit `<wsdlUrl>`, you must also provide a unique value to `<name>` in the Envoy.

Edit Connector config

If you configure AP Invoice eForm to work with Perceptive Connector for Dynamics AX 4.0.x, you may require to edit the default value to provide the baseUrl to the Dynamics AX connector endpoint hosted in PCR. To edit the value, complete the following steps.

1. Navigate to **[*drive*:]\<*INServer*>\temp\apef_install\etc**.
2. Open the **AP_Config.xml** file.
3. Under the `<connectorconfig>` element, edit the value for `<baseUrl>`.

Note Typical format is <baseUrl>http://localhost:80/rs/DaxConnector</baseUrl>.

Appendix A: AP Invoice eForm virtual tables

This appendix lists each possible AP Invoice eForm virtual table and defines the required CSV file structure. This section identifies each required table, in alphabetical order, as well as any required columns within the table.

Account Assignment Lines

The ACC_ASSMT_LINE table contains an entry for each account assignment line associated with an invoice line for blanket POs and service POs in SAP.

	A	B	C	D	E	F	G	H
1	**TableName***	**PO Number***	**PO Line Number**	**PO Item Number***	**Serial Number***	**GL Account**	**Cost Center**	**Profit Center**
2	ACC_ASSMT_LINE	650000109	1	B100	01	400000	30821	30811
3	ACC_ASSMT_LINE	650000109	1	B100	02	400001	30822	30812
4	ACC_ASSMT_LINE	650000109	2	B200	01	400000		30811
5	ACC_ASSMT_LINE	650000109	2	B200	02	400001		30812

	I	J	K	L	M	N	O	P	Q
1	**WBS Element**	**Fund**	**Funds Center**	**Business Area**	**Cont Area**	**VAT Jur**	**VAT Code**	**Quantity**	**Dist Percent**
2		251	28000	9900	2000	LA000000	I1	700.00	70.00
3		252	28001	9910	3000	LB000000	I2	300.00	30.00
4	3-4200/215	251	28000	9900	2000	LA000000	I1	500.00	50.00
5	E4050	252	28001	9910	3000	LB000000	I2	500.00	50.00

*Indicates a required column. Each combination of required columns must be unique. Perceptive Content uses these values, separated by a dash, to create the project or folder name for the record.

AOC GL Account

The AOC_GL_ACCT table contains an entry for each add on cost (AOC) code and general ledger (GL) account code combination, for use with distributed additional amounts. This table stores the AOC code and optional default values for the business unit, accounting unit, and up to six additional GL Account fields.

When using AOC GL account codes in the eForm, this table provides the default values for populating the GL Distributions section for purchase order invoices. In the eForm, the user may manually key in alternate values.

You can set a default AOC code in the AP_Skin.xml file for the Sales Tax, Freight, Misc., and Additional Amounts <*1-8*> fields. A default AOC code is not required.

Notes

- The combination of GL Account fields (BusUnitID through GLAcct8) in this table should correspond with a matching entry in the GL Account table with the same field names.

- The table inserts a zero as the placeholder for any enabled, but blank, GL Account fields.

	A	B	C	D	E	F	G
1	**TableName***	**Activ**	**AOCCode***	**AOCDesc.**	**BusUnitID**	**AcctgUnit**	**GLAcct**
2	AOC_GL_ACCT	Y	FR	Freight	1234	10100	17900
3	AOC_GL_ACCT	Y	FS	Fuel Surcharge	1234	10100	22500
4	AOC_GL_ACCT	Y	HA	Handling	1234	10100	22500
5	AOC_GL_ACCT	Y	IN	Insurance	1234	10100	57020

	H	I	J	K	L	M
1	**GLSubAcct**	**GLAcctDesc**	**GLAcct5**	**GLAcct6**	**GLAcct7**	**GLAcct8**
2	300	Freight, Express & Postage			200	
3	100	Fuel/Diesel Oil				2008
4	200	Miscellaneous	100		200	
5	100	Insurance	200			2008

*Indicates a required column. Each combination of required columns must be unique. Perceptive Content uses these values, separated by a dash, to create the project or folder name for the record.

AOC GL Code

The AOC_GL_CODE table contains an entry for each add on cost (AOC) code and up to 24 optional GL Codes and descriptions.

When using AOC GL codes in AP Invoice eForm, this table provides the default values for populating the GL Distributions section for purchase order invoices. In the eForm, the user may manually key in alternate values.

You can set the default AOC code in the AP_Skin.xml file for the Sales Tax, Freight, Misc, and Additional Amounts <*1-8*> fields. A default AOC code is not required.

Note Each GL Code value in this table (GLCode1 - GLCode24) should correspond with a matching entry in the GL Code table. For example, if the AOC_GL_CODE table has a value of 200 for GLCode1, then the GLCODE table should have an entry with GLNumber = 1 and GLCode = 200.

	A	B	C	D	E	F
1	**TableName***	**Active***	**AOCCode***	**AOCDesc**	**GLCode1**	**GLCodeDesc1**
2	AOC_GL_ CODE	Y	FR	Freight	100	Company A
3	AOC_GL_ CODE	Y	FS	Fuel Surcharge	100	Company A
4	AOC_GL_ CODE	Y	HA	Handling	200	Company B
5	AOC_GL_ CODE	Y	IN	Insurance	200	Company B

	G	H	I	J	K	L
1	**GLCode 2**	**GLCodeDesc2**	**GLCode3**	**GLCodeDesc3**	**GLCode<*4-24*>**	**GLCode<*4-24*>**
2	1105	Freight, Express & Postage	240006	Misc Expense	…	…
3	1106	Fuel/Diesel Oil	240007	Service	…	…
4	1107	Miscellaneous			…	…
5	1108	Insurance	240006	Misc Expense	…	…

*Indicates a required column. Each combination of required columns must be unique. Perceptive Content uses these values, separated by a dash, to create the project or folder name for the record.

Business Unit – *required*

The required BUS_UNIT table contains an entry for every business unit, company, or entity.

- The VendorGroup field corresponds to the VendorGroup field in the Vendor table and allows you to associate a group of vendors with each business unit.
- If all business units share the same vendor pool, use ALL as the value.

Value added tax (VAT)

For solutions that implement value added tax (VAT), review the following information.

- To provide a default Jurisdiction or VAT registration number (VAT ID) value for each business unit, populate the Jurisdiction and VAT ID fields.
- If you do not provide a default and leave the value blank, the user can manually key the Jurisdiction and VAT ID values into the eForm as necessary.
- For an overview of incorporating VAT into the eForm, refer to the "Configure value added tax within the eForm" section in the *Perceptive AP Invoice eForm Advanced Design and Setup Guide*.

	A	B	C	D	E	F	G
1	**TableName***	**Active***	**BusUnitID***	**BusUnitName***	**VendorGroup***	**Jurisdiction**	**VAT ID**
2	BUS_UNIT	Y	1234	ACME	ACME	US	
3	BUS_UNIT	Y	4500	CGI Europe	EUR	FR	FRXX999999999
4	BUS_UNIT	Y	9001	International	EUR	FR	FR1234567890
5	BUS_UNIT	Y	4321	LGE Corporation	EUR	DE	DE123456789
6	BUS_UNIT	Y	9000	UK Retail Ltd	UKLTD	GB	GB999 9999 73

*Indicates a required column. Each combination of required columns must be unique. Perceptive Content uses these values, separated by a dash, to create the project or folder name for the record.

Currency

The optional CURRENCY table contains an entry for each currency code, by Business Unit.

The eForm uses the optional Variance Amount and Variance Percent columns to determine tolerance levels when validating value added tax (VAT) totals. If specifying a tolerance level, you can only use one of the two columns per currency. If you do not specify a variance amount or percent and you enable validation, the Vat Total must match the sum of the line item VAT exactly.

Using this table as an example:

- The Variance Amount on British Pounds is .05. If a user processes an invoice with British Pounds as the currency, and the invoice VAT Total Amount is £100, then the VAT Total in the summary section must be between £99.95 and £100.05 to pass validation.
- The Variation Percent on Euros is 1.5%. If a user processes an invoice with Euros as the currency, and the VAT Total Amount is €10, then the VAT Total in the summary must be between €9.85 and €10.15 to pass validation.

Notes

- The BusUnitID field corresponds to the BusUnitID field in the Business Unit table.
- The PayGroup field is optional. If used, it exports with invoice information.

	A	B	C	D	E	F	G	H
1	**TableName***	**BusUnitID***	**CurrencyCode***	**CurrencyName***	**PayGroup**	**Active***	**Variance Amount**	**Variance Percent**
2	CURRENCY	1234	USD	US Dollar	7000	Y		
3	CURRENCY	1234	GBP	Pound	8100	Y	.05	
4	CURRENCY	1234	EUR	Euro	7000	Y		1.5
5	CURRENCY	1234	MXN	Peso		Y	12.5	

*Indicates a required column. Each combination of required columns must be unique. Perceptive Content uses these values, separated by a dash, to create the project or folder name for the record.

Currency Country

The COUNTRYCURRENCY table contains an entry for each unique country and currency pair. It ties each pair to a locale for number and percentage formatting. This table is optional. However, if the table is blank, the eForm follows the language formatting rules for the operating system locale.

Notes

- The CurrencyCode field corresponds to the CurrencyCode field in the Currency and Vendor tables.
- The LocaleCode field requires the 2-character or 4-character language and country codes available for the supported languages. For more information, refer to the “Configure AP Invoice eForm for a global environment” section of the *Perceptive AP Invoice eForm Advanced Design and Setup Guide*.

	A	B	C	D	E
1	**TableName***	**CurrencyCode***	**CountryCode***	**LocaleCode***	**Active***
2	COUNTRYCURRENCY	USD	US	en-US	Y

	A	B	C	D	E
3	COUNTRYCURRENCY	EUR	FR	en-FR	Y
4	COUNTRYCURRENCY	EUR	DE	de-DE	Y
5	COUNTRYCURRENCY	BRL	BR	pt-BR	Y

*Indicates a required column. Each combination of required columns must be unique. Perceptive Content uses these values, separated by a dash, to create the project or folder name for the record.

GL Account - *required*

The GL_ACCT table includes an entry for every general ledger account code combination. This table represents one option for coding non-PO invoices. The other option is to use the GL Code table.

- This table supports two to six elements.
- The eForm requires this table for non-purchase order processing, if you do not use the GL Code table.
- This table provides combination validation, without the need for custom development. Implementations with Lawson often use this table.
- The table inserts a zero as the placeholder for any enabled, but blank, GL Account fields.

	A	B	C	D	E	F	G
1	**TableName***	**Active***	**BusUnitID**	**AcctgUnit**	**GLAcct**	**GLSubAcct**	**GLAcctDesc**
2	GL_ACCT	Y	1234	101	17900	100	Office Supplies - Misc.
3	GL_ACCT	Y	1234	101	17900	200	Office Supplies - Copiers
4	GL_ACCT	Y	1234	101	17900	300	Office Supplies - Printers
5	GL_ACCT	Y	1234	101	22500	100	Maintenance – Buildings
6	GL_ACCT	Y	1234	101	22500	200	Maintenance – Grounds
7	GL_ACCT	Y	1234	101	22500	100	Maintenance – HVAC

	H	I	J	K
1	**GLAcct5**	**GLAcct6**	**GLAcct7**	**GLAcct8**
2	3370	7071	39000	
3	3370	7071	39000	
4	3370	7071	39000	
5	3375	7072	45001	10000
6	3375	7072	45001	10001
7	3375	7072	45001	10002

*Indicates a required column. Each combination of required columns must be unique. Perceptive Content uses these values, separated by a dash, to create the project or folder name for the record.

GL Code - *required*

The GLCODE table includes an entry for every general ledger code. This table represents one option for use in coding non-PO invoices. The other option is to use the GL Account table.

- This table supports one to 24 elements. Define each number with the GL Number value.
- The eForm requires this table for non-purchase order processing, if the GL Account table is not used.
- The GLNumber column groups a set of values, which you define in the GLCode column. You use these GLNumber sets to populate lists in the GL Distribution section of the eForm user interface. In AP_Skin.xml, in the `<glline>` elements, you refer to the GLNumber set with the `source` attribute. Using this example table, if you wanted the GL1 field in the eForm to display a list with the values 100 and 200 as options, you would set the `source` to `1`.

```
<glline label="lblGL1" field="txtGL1" showdesc="true" searchable="true"
freeform="false" source="1" line="1" typeaheadchars="2" visible="true"
disabled="false" required="true" width="50px" captionkey="LBL_GLLINE_GL1"
descriptionkey="LBL_GLLINE_GL1_DESC"/>
```

  Using this table again, if you wanted the GL field in the eForm to display a list with the values 1101, 1102, 1103, and 1104, you would set the `source` to `2`.

- Constraint1 and 2, in conjunction with the GL User table, control which GL codes are available for specific users. If you do not use constraints to limit the GL codes per user, both Constraint1 and Constraint2 values should be set to ALL.

	A	B	C	D	E	F	G	
1	**TableName***	**Constraint1***	**Constraint2***	**GLNumber***	**GLCode***	**Description**	**Active***	
2	GL CODE	ALL	ALL	1	100	Company A	Y	
3	GL CODE	ALL	ALL	1	200	Company B	Y	
4	GLCODE	Department_01	KCMO	2	1101	Petty Cash	Y	
5	GLCODE	Department_01	KCMO	2	1102	Travel	Y	
6	GLCODE	ALL	ALL	2	1103	Livestock	Y	
7	GLCODE	ALL	ALL	2	1104	Laptops	Y	
8	GLCODE	ALL	ALL	3	240006	Acct Unit 1	Y	
9	GLCODE	ALL	ALL	3	240007	Acct Unit 2	Y	

*Indicates a required column. Each combination of required columns must be unique. Perceptive Content uses these values, separated by a dash, to create the project or folder name for the record.

GL User

The GLUSER table includes an entry for every general ledger user. It provides the values used to filter the GL Code list described above and is optional. Each GL User has access to GL Codes based on the constraints defined in this table.

Notes

- If you need to limit GL codes on a user-by-user basis, use this table. If all entries in the GLCODE table have constraints set to ALL, the GLUSER table is not necessary.
- The UserName field corresponds to an existing Perceptive Content User Name and is case-sensitive. You must explicitly define all users with restricted GL Code access in this table. This table does not support groups.

	A	B	C	D	E
1	**TableName***	**UserName***	**Constraint1**	**Constraint2**	**Active**
2	GLUSER	JDOE	Department_01	KCMO	Y
3	GLUSER	JJONES	Department_05	KCMO	Y

*Indicates a required column. Each combination of required columns must be unique. Perceptive Content uses these values, separated by a dash, to create the project or folder name for the record.

Location - *required*

The LOCATION table contains an entry for every tax jurisdiction or geocode between which an invoice is paid. It then sets the default Sales Use Tax (SUT) code and Ultimate Use Tax (UUT) codes for that location.

- The table reserved the SetID field for future use. The current value is always SHARE.
- The SUTCode field corresponds to the TaxCode field in the SUT Code table.

	A	B	C	D	E	F	G
1	**TableName***	**SetID***	**LocCode***	**LocName***	**SUTCode**	**UUTCode**	**Active***
2	LOCATION	SHARE	KC01	CARTER PLANT – KC	10100	A02	Y
3	LOCATION	SHARE	DEN01	SMITH WHSE - DENVER	15600	B14	Y

*Indicates a required column. Each combination of required columns must be unique. Perceptive Content uses these values, separated by a dash, to create the project or folder name for the record.

Payment Terms

The PAYMENTTERMS table calculates the voucher payment due date based on the vendor's payment terms criteria and calculation method.

The PaymentTermsCode field corresponds to the PaymentTerms field in the Vendor table.

The eForm calculates the payment due date based on the formula you designate in the PaymentCalcMethod field and the value you define in the PaymentDay field. The options include:

- **FROM_INV_DT** = Calculate due date based on invoice date

  Due Date = Invoice Date + Number of Days [PaymentDay value]

- **FROM_EOM** = Calculate due date based on end of the month date

  Due Date = Last day of the month, based on the invoice date + Number of Days [PaymentDay value]

- **DAY_CURR** = Due date is a specified day of the current month

  Due Date = Day of the month [PaymentDay value] for the current month (based on invoice date)

- **DAY_NEXT** = Due date is a specified day of the next month

  Due Date = Day of the month [PaymentDay value] for the following month (based on invoice date)

	A	B	C	D	E
1	**TableName***	**PaymentTermCode***	**PaymentCalcMethod***	**PaymentDay***	**Active***
2	PAYMENTTERMS	Net 10	FROM_INV_DT	10	Y
3	PAYMENTTERMS	Net 30	FROM_INV_DT	30	Y
4	PAYMENTTERMS	Fixed 15	DAY_CURR	15	Y

*Indicates a required column. Each combination of required columns must be unique. Perceptive Content uses these values, separated by a dash, to create the project or folder name for the record.

Purchase Order - *required*

The PO table contains an entry for every open purchase Order. The eForm requires this table to process purchase order-based invoices.

Notes

- The VendorID and RemitTo fields correspond to the VendorID and RemitTo fields in the Vendor table.
- The PODate field is the purchase order creation or issue date.
- The CloseDate field is the date the user marked purchase order fully received and paid. After a line indicates a CloseDate, the PO number is no longer available to users in the eForm.

	A	B	C	D	E	F	G	H	I
1	**Table Name**	**BusUnitID ***	**PONumber***	**VendorID ***	**RemitTo***	**PO Amount***	**Currency**	**PODate**	**CloseDate**
2	PO	1234	650000109	4500	2	127.30	USD	20130615	
3	PO	1234	650000110	4500	1	158.17	USD	20130609	20130621
4	PO	1234	650000201	4500	2	584.17	EUR	20130611	
5	PO	1234	650001001	1000	1	7300.8	GBP	20130612	
6	PO	1234	650001012	1000	1	538.80	EUR	20130619	
7	PO	1234	650000920	4500	1	1150.00	CAD	20130620	

*Indicates a required column. Each combination of required columns must be unique. Perceptive Content uses these values, separated by a dash, to create the project or folder name for the record.

Purchase Order Line - *required*

The PO_LINE table contains a line item entry for every open purchase order. The eForm requires this table to process purchase order-based invoices.

Notes

- The BusUnitID field corresponds to the BusUnitID field in the Business Unit table.
- The Jurisdiction field corresponds to the Jurisdiction field in the Business Unit table.
- The POItemQty column reflects one of the following states, depending on the matching type:

  2-way: quantity ordered. The PO_LINE table initially loads with the quantity ordered on the original purchase order. The POItemQty does not reduce as the vendor or supplier invoices each line.

  3-way: quantity received but not invoiced. The PO_LINE table loads with quantity received at the time of receipt. The total quantity reduces as the vendor or supplier invoices each line. Assuming that you continuously update the PO_LINE table data, the field only reflects open purchase order quantities. You can achieve an indirect 3-way match using AP Invoice eForm or a direct 3-way match if the user performs the matching in the ERP.

4-way: quantity received in good condition but not invoiced. The PO_LINE table loads with quantity received at the time of receipt. As with 3-way matching, the quantity reduces as the vendor or supplier invoices each line and the field only reflects open purchase order quantities. You can achieve an indirect 4-way match using AP Invoice eForm or a direct 4-way match if the user performs the matching in the ERP.

Note Best practice is to use either 3-way or 4-way matching.

- The CloseDate field is the date the ERP marks the purchase order as fully received and matched. After a line indicates a CloseDate, it is no longer available to users for matching in the eForm.
- Field1 to Field10 are optional fields used for storing additional PO information. Display these values on the eForm by activating the appropriate value in the eForm.

Value added tax (VAT)

For solutions that implement value added tax (VAT), review the following information.

To provide a default Jurisdiction and VAT code for each purchase order line, populate the Jurisdiction and VATCode fields. If left blank, the user can manually key the Jurisdiction and select a VAT code in the eForm, as necessary. For an overview of incorporating VAT into the eForm, refer to the “Configure value added tax within the eForm” section of the *Perceptive AP Invoice eForm Advanced Design and Setup Guide*.

	A	B	C	D	E	F	G	H	I	J
1	**Table Name***	**Bus Unit ID***	**PO Number***	**PO Line Numbe**	**PO Item Number***	**PO Material Number**	**PO Item Desc***	**PO Item Quantity ***	**PO Item UOM***	**PO Item Unit Price***
2	PO_LINE	1234	650000109	1	B100	DPC1003	Binder Clips	20	BOX	2.49
3	PO_LINE	1234	650000110	2	CPW 16X	DPC1006	Copier Paper 200	5	BOX	15.50
4	PO_LINE	1234	650000201	1	AAG-Desk	DPC1024	At-A-Glace Desk Pack	10	BOX	6.39
5	PO_LINE	1234	650001001	2	PPCAC		Plastic Paper Clips	20	BOX	3.19
6	PO_LINE	1234	650001012	3	TTD114	DPC1088	Things to Do Pads	5	BOX	3.89
7	PO_LINE	1234	650000920	1	1-PN		10 Penny Nails	10	BOX	1.88

	K	L	M	N	O	P	Q	R	S
1	**PO Item Ext Amt***	**Taxable**	**Date Close**	**Jurisdiction**	**VAT Code**	**GR Document**	**GR Fiscal**	**GR Item**	**PO Item Category**
2	49.80	Y		FR	V0	GRDocument	2011	GR_Item1	0
3	77.5	Y		FR	V1	GRDocument	2011	GR_Item2	0
4	63.9	N		FR	V2	GRDocument	2011	GR_Item3	1
5	63.8	N		FR	V3	GRDocument	2011	GR_Item4	0
6	19.45	Y		FR	V4	GRDocument	2011	GR_Item5	1
7	18.8	Y		FR	V5	GRDocument	2011	GR_Item6	1

*Indicates a required column. Each combination of required columns must be unique. Perceptive Content uses these values, separated by a dash, to create the project or folder name for the record.

Sales Use Tax (SUT) Codes

The SUT_CODE table contains an entry for all the Sales Use Tax (SUT) codes.

	A	B	C
1	**TableName***	**Active***	**TaxCode***
2	SUT_CODE	Y	10100
3	SUT_CODE	Y	11200

*Indicates a required column. Each combination of required columns must be unique. Perceptive Content uses these values, separated by a dash, to create the project or folder name for the record.

Sales Use Tax (SUT) Apply Codes

The SUT_APPLY table contains an entry for the status of the SUT code application. It indicates whether to apply the SUT code to the current invoice.

	A	B	C
1	**TableName***	**Active***	**ApplyCode***
2	SUT_APPLY	Y	YES
3	SUT_APPLY	Y	NO

*Indicates a required column. Each combination of required columns must be unique. Perceptive Content uses these values, separated by a dash, to create the project or folder name for the record.

Special Handling

The SPEC_HANDLING table contains an entry for each special handling instruction that the user can apply to the current invoice. The options in this table appear in AP Invoice eForm, in the header section, in the Special Handling list.

	A	B	C	D
1	**TableName***	**Activ**	**SpecHandlingCod**	**SpecHandlingDesc***
2	SPEC_HANDLING	Y	EXP	Expedite Payment
3	SPEC_HANDLING	Y	HOLD	Hold for Supplier Pickup
4	SPEC_HANDLING	Y	ATT	Attach Supporting Documentation

*Indicates a required column. Each combination of required columns must be unique. Perceptive Content uses these values, separated by a dash, to create the project or folder name for the record.

VAT Code

The VAT_CODE table provides a mapping between the Jurisdiction, the VAT Code and the VAT Rate. Each rate requires a combination of Jurisdiction and VAT Code. Jurisdiction is not optional, but you may use a default value if the Jurisdiction never varies. For example, if your solution only processes local invoices, you could provide a single default value throughout the Jurisdiction column. Provide VAT rates in the table as a percentage between 0 and 100. For example, enter 7.5% as 7.5, rather than .75.

	A	B	C	D	E
1	**TableName***	**Jurisdiction***	**VATCode***	**VATRate***	**Description**
2	VAT_CODE	FR	V0	0	France zero tax rate
3	VAT_CODE	FR	V1	19.6	France full tax rate
4	VAT_CODE	FR	V2	7	France reduced tax rate
5	VAT_CODE	FR	V7	0	France tax exempt, foreign vendor

*Indicates a required column

VAT Registration Number (VAT ID) Vendor

The VAT_ID_VENDOR table contains an entry for each vendor VAT registration number (VAT ID) and provides the ability to relate one or more VAT IDs to each vendor.

Notes

- The BusUnitID field corresponds to the BusUnitID field in the Business Unit table.
- The VendorID field corresponds to the VendorID field in the Vendor table.

	A	B	C	D	E
1	**TableName***	**Active***	**BusUnitId***	**VendorID***	**VATID***
2	VAT_ID_VENDOR	Y	4500	113000	FR1X624486189
3	VAT_ID_VENDOR	Y	4500	113000	FR5232587290
4	VAT_ID_VENDOR	Y	4321	114000	DE947362067
5	VAT_ID_VENDOR	Y	4321	114001	DE295091634
6	VAT_ID_VENDOR	Y	9000	115000	GB346 5860 73

*Indicates a required column

VAT Registration Number (VAT ID) Business Unit

The VAT_ID_BUS_UNIT table contains an entry for each business unit's VAT registration number (VAT ID) and provides the ability to relate one or more VAT IDs to each business unit.

Note

- The BusUnitID field corresponds to the BusUnitID field in the Business Unit table.

	A	B	C	D
1	**Table Name***	**Active***	**BusUnitID***	**VATID***
2	VAT_ID_BUS_UNIT	Y	4500	FRXX999999999
3	VAT_ID_BUS_UNIT	Y	4500	FR999999999
4	VAT_ID_BUS_UNIT	Y	9001	FR1234567890
5	VAT_ID_BUS_UNIT	Y	4321	DE123456789
6	VAT_ID_BUS_UNIT	Y	4321	DE234567891
7	VAT_ID_BUS_UNIT	Y	9000	GB999 9999 73

*Indicates a required column. Each combination of required columns must be unique. Perceptive Content uses these values, separated by a dash, to create the project or folder name for the record.

Vendor - *required*

The VENDOR table contains an entry for every Vendor.

Notes

- The VendorGroup field corresponds to the VendorGroup field in the Business Unit table and allows you to associate a group of vendors with each Business Unit. If all Business Units share the same vendor pool, use ALL as the value.
- The combination of VendorID and RemitTo make up the unique vendor number. The RemitTo value is an address sequence number.
- The PaymentTerms field corresponds to the PaymentTermCode in the Payment Terms table. The value specified for each vendor is the default value.
- The VATID field stores the tax identification number, which includes the TIN number for US Vendors and VAT Registration number for European vendors. The value specified for each vendor is the default value.
- Field1 and Field2 are optional fields used for storing additional vendor information. Display these values on the eForm by activating the appropriate value in the eForm.

Value added tax (VAT)

For solutions that implement value added tax (VAT), review the following information.

To provide a default VAT registration number (VAT ID) for each vendor, populate the VAT field. If left blank, the user can manually key a VAT registration number into the eForm. For an overview of incorporating VAT into the eForm, refer to the “Configure value added tax within the eForm” section in the *Perceptive AP Invoice eForm Advanced Design and Setup Guide*.

	A	B	C	D	E	F	G	H
1	**Table Name***	**Active***	**Vendor Group***	**VendorID***	**Remit To***	**VendorName***	**Address1**	**Address2**
2	VENDOR	Y	ACME	110001	1	General Supply Co	1600 Main St, B	
3	VENDOR	Y	ACME	110002	2	General Supply Co	2626 Broadway Ave	Suite 100
4	VENDOR	Y	EUR	113000	1	Le Zèbre	2 Avenue Gabriel	

	I	J	K	L	M	N	O	P	Q
1	**City**	**State**	**Postal Code**	**Country Code**	**VendorShortName**	**Payment Terms**	**VATID**	**Field1**	**Field2**
2	Chicago	IL	60600	US	GENSUP	Net 30			
3	New York	NY	86888	US	GENSUP	Net 30			
4	Paris		75008	FR	LZEBRE	Net 45	FR1X123456		

*Indicates a required column. Each combination of required columns must be unique. Perceptive Content uses these values, separated by a dash, to create the project or folder name for the record.

Vendor Location

The VENDOR_LOCATION table contains one or more location values that relate to the selected vendor. PeopleSoft implementations most commonly use this table.

Note

- The VendorID field corresponds to the VendorID field in the Vendor table.

	A	B	C	D	E
1	**TableName***	**VendorID***	**VendorLocCode***	**VendorLocName***	**Active***
2	VENDOR_LOCATION	11001	P	PRIMARY	Y
3	VENDOR_LOCATION	11002	P	PRIMARY	Y
4	VENDOR_LOCATION	11003	P	PRIMARY	Y
5	VENDOR_LOCATION	11003	S	SECONDARY	Y

*Indicates a required column. Each combination of required columns must be unique. Perceptive Content uses these values, separated by a dash, to create the project or folder name for the record.

Appendix B: Reserved characters in XML and forms

We do not recommend using special characters in queue names.

XML reserves some characters commonly used as markup delimiters. These reserved characters cannot appear as a literal string in XML character data, such as the text value of an element.

Sometimes it is necessary to display XML reserved characters as text. To do this, you must use an escape sequence. XML provides standard escape sequences, called character entity references, for these reserved characters. These entity references provide a way to refer to a character that is not universally encodable.

The following table illustrates these reserved characters and their replacement entity references.

Reserved Character	Meaning	Entity Reference/ Escape Sequence	Comments/What to use when coding a form
>	Greater than	$gt;	For compatibility, this character must be escaped using the entity reference or as a character reference when it appears in the string "]]>" in content and when that string is not marking the end of a CDATA section.
<	Less than	<	This character should always appear in content as an escape character so that it does not interact with the syntax of the markup.
&	Ampersand	&	This character should always appear in content as an escape character so that it does not interact with the syntax of the markup.
'	Apostrophe	'	You can symbolize the apostrophe or single-quote character (') with this character entity reference when you need to embed a single-quote or apostrophe inside a string that is already single-quoted. Although ' is part of the XML language, it is not defined in HTML. Use ' if text is passed to an HTML user agent.
"	Quotation mark	"	You can symbolize the double-quote character (") with this character entity reference when you need to embed a double-quote inside a string that is already double-quoted. **Note** Use \" when writing unparsed data to XML.
%	Percent	%	The Notification Services XML vocabulary reserves the percent sign (%) for denoting parameters. **Note** &#37 is an escape sequence, but not an entity reference defined by XML.
\	Slash	None required	**Note** Use \\ when writing unparsed data to XML.

Appendix C: Functions and associated parameters of CustomConnector.js

The Custom Connector is an iScript file that provides you a framework for developing your own connector. In the CustomConnector.js file, you can write code that leverages the parameters associated with the available functions to customize the connector and perform lookup operations from any data source, such as an external database or remote web service. The output parameters of each function in the CustomConnector must be identical to the parameters listed in the CSV file structure of AP Invoice eForm virtual tables. To view identical parameters in AP Invoice eForm's virtual tables, refer to Appendix A: AP Invoice eForm virtual tables.

Customize functions and associated parameters

The following list provides information about the functions available in the CustomConnector file in the order they appear within the file.

initialize

This function provides an initialization hook in the connector interface. The function may be implemented to contain database initialization code.

Code syntax

```
CustomConnector.prototype.initialize = function()
```

finalize

This function provides an exit hook in the connector interface. The function may be implemented to contain database teardown code.

Code syntax

```
CustomConnector.prototype.finalize = function()
```

getBusinessUnits

This function returns an array of all business units to the AP Invoice eForm through the abstraction layer. The function is invoked when you run AP_Maintenance.js script or when the cache expires and the returned list is cached in AP_GetCompaniesCache.js file.

The following input parameters are passed to the function.

- **moreResults**. An indicator to denote whether more results are available or not. This value is passed by reference. If the complete list is not returned, set the value to `true`. The default value is `false`. (**Data type**: boolean)
- **maxResults**. The value configured in AP_Config.xml for the maximum results to return from the data source. This parameter can be used to query the data source to restrict results. However, a query to restrict the results is not mandatory as the abstraction layer restricts the results based on this value. (**Data type**: number)

The function returns an array of objects with the following properties.

- **Code**. Business unit code (**Data type**: string; **Requirement**: Mandatory)
- **Name**. Name of the business unit (**Data type**: string; **Requirement**: Mandatory)
- **VendorGroup**. Vendor group of the business unit (**Data type**: string; **Requirement**: Optional)
- **Jurisdiction**. Jurisdiction of the business unit (**Data type**: string; **Requirement**: Optional)
- **VAT**. VAT code of the business unit (**Data type**: string; **Requirement**: Optional)

Code syntax

```
CustomConnector.prototype.getBusinessUnits = function(&moreResults, maxResults)
```

getVATCodes

This function returns an array of all VAT codes to the AP Invoice eForm through the abstraction layer. The function is invoked when you run AP_Maintenance.js script or when the cache expires and the returned list is cached in AP_GetCompaniesCache.js file.

The following input parameters are passed to the function.

- **moreResults**. An indicator to denote whether more results are available or not. This value is passed by reference. If the complete list is not returned, set the value to `true`. The default value is `false`. (**Data type**: boolean)
- **maxResults**. The value configured in AP_Config.xml for the maximum results to return from the data source. This parameter can be used to query the data source to restrict results. However, a query to restrict the results is not mandatory as the abstraction layer restricts the results based on this value. (**Data type**: number)

The function returns an array of objects with following properties.

- **Code**. VAT code (**Data type**: string; **Requirement**: Mandatory)
- **Jurisdiction**. Jurisdiction corresponding to the VAT code (**Data type**: string; **Requirement**: Optional)
- **Rate**. VAT rate percentage (**Data type**: string; **Requirement**: Optional)
- **Purpose**. Description of the VAT code (**Data type**: string; **Requirement**: Optional)

Code syntax

```
CustomConnector.prototype.getVATCodes = function(&moreResults, maxResults)
```

getLocations

This function returns an array of all locations to the AP Invoice eForm through the abstraction layer. The function is invoked when you run AP_Maintenance.js script or when the cache expires and the returned list is cached in AP_GetCompaniesCache.js file.

The following input parameters are passed to the function.

- **moreResults**. An indicator to denote whether more results are available or not. This value is passed by reference. If the complete list is not returned, set the value to `true`. The default value is `false`. (**Data type**: boolean)
- **maxResults**. The value configured in AP_Config.xml for the maximum results to return from the data source. This parameter can be used to query the data source to restrict results. However, a query to restrict the results is not mandatory as the abstraction layer restricts the results based on this value. (**Data type**: number)

The function returns an array of objects with the following properties.

- **Code**. Location code (**Data type**: string; **Requirement**: Mandatory)
- **Name**. Location name (**Data type**: string; **Requirement**: Mandatory)
- **SetID**. Set ID of the location (**Data type**: string; **Requirement**: Optional)
- **SUTCode**. Sales use tax code (**Data type**: string; **Requirement**: Optional)
- **UUTCode**. UUT code (**Data type**: string; **Requirement**: Optional)

Code syntax

```
CustomConnector.prototype.getLocations = function(&moreResults, maxResults)
```

getSUTCodes

This function returns an array of all SUT codes to the AP Invoice eForm through the abstraction layer. The function is invoked when you run AP_Maintenance.js script or when the cache expires and the returned list is cached in AP_GetCompaniesCache.js file.

The following input parameters are passed to the function.

- **moreResults**. An indicator to denote whether more results are available or not. This value is passed by reference. If the complete list is not returned, set the value to `true`. The default value is `false`. (**Data type**: boolean)
- **maxResults**. The value configured in AP_Config.xml for the maximum results to return from the data source. This parameter can be used to query the data source to restrict results. However, a query to restrict the results is not mandatory as the abstraction layer restricts the results based on this value. (**Data type**: number)

The function returns an array of objects with the following properties.

- **Code**. SUT code (**Data type**: string; **Requirement**: Mandatory)
- **Name**. SUT name (**Data type**: string; **Requirement**: Optional)

Code syntax

```
CustomConnector.prototype.getSUTCodes = function(&moreResults, maxResults)
```

getSUTApplyCodes

This function returns an array of all SUT Apply codes to the AP Invoice eForm through the abstraction layer. The function is invoked when you run AP_Maintenance.js script or when the cache expires and the returned list is cached in AP_GetCompaniesCache.js file.

The following input parameters are passed to the function.

- **moreResults**. An indicator to denote whether more results are available or not. This value is passed by reference. If the complete list is not returned, set the value to `true`. The default value is `false`. (**Data type**: boolean)
- **maxResults**. The value configured in AP_Config.xml for the maximum results to return from the data source. This parameter can be used to query the data source to restrict results. However, a query to restrict the results is not mandatory as the abstraction layer restricts the results based on this value. (**Data type**: number)

The function returns an array of objects with the following properties.

- **Code**. SUT Apply code (**Data type**: string; **Requirement**: Mandatory)
- **Name**. SUT Apply name (**Data type**: string; **Requirement**: Optional)

Code syntax

```
CustomConnector.prototype.getSUTApplyCodes = function(&moreResults, maxResults)
```

getCurrencyCodes

This function returns an array of all currency codes to the AP Invoice eForm through the abstraction layer. The function is invoked when you run AP_Maintenance.js script or when the cache expires and the returned list is cached in AP_GetCompaniesCache.js file.

The following input parameters are passed to the function.

- **moreResults**. An indicator to denote whether more results are available or not. This value is passed by reference. If the complete list is not returned, set the value to `true`. The default value is `false`. (**Data type**: boolean)
- **maxResults**. The value configured in AP_Config.xml for the maximum results to return from the data source. This parameter can be used to query the data source to restrict results. However, a query to restrict the results is not mandatory as the abstraction layer restricts the results based on this value. (**Data type**: number)

The function returns an array of objects with the following properties.

- **BusUnit**. Business unit of the currency code (**Data type**: string; **Requirement**: Mandatory)
- **CurrCode**. Currency code (**Data type**: string; **Requirement**: Optional)
- **CurrName**. Description of the currency (**Data type**: string; **Requirement**: Optional)
- **PayGroup**. Pay group of the currency (**Data type**: string; **Requirement**: Optional)
- **VarAmount**. Variance amount (**Data type**: string; **Requirement**: Optional)
- **VarPercent**. Variance percentage (**Data type**: string; **Requirement**: Optional)

Code syntax

```
CustomConnector.prototype.getCurrencyCodes = function(&moreResults, maxResults)
```

getSpecHandlingCodes

This function returns an array of all spec handling codes to the AP Invoice eForm through the abstraction layer. The function is invoked when you run AP_Maintenance.js script or when the cache expires and the returned list is cached in AP_GetCompaniesCache.js file.

The following input parameters are passed to the function.

- **moreResults**. An indicator to denote whether more results are available or not. This value is passed by reference. If the complete list is not returned, set the value to `true`. The default value is `false`. (**Data type**: boolean)
- **maxResults**. The value configured in AP_Config.xml for the maximum results to return from the data source. This parameter can be used to query the data source to restrict results. However, a query to restrict the results is not mandatory as the abstraction layer restricts the results based on this value. (**Data type**: number)

The function returns an array of objects with the following properties.

- **Code**. Special handling code (**Data type**: string; **Requirement**: Mandatory)
- **Name**. Description of the special handling code (**Data type**: string; **Requirement**: Optional)

Code syntax

```
CustomConnector.prototype.getSpecHandlingCodes = function(&moreResults, maxResults)
```

getPaymentTermsCodes

This function returns an array of all payment terms to the AP Invoice eForm through the abstraction layer. The function is invoked when you run AP_Maintenance.js script or when the cache expires and the returned list is cached in AP_GetCompaniesCache.js file.

The following input parameters are passed to the function.

- **moreResults**. An indicator to denote whether more results are available or not. This value is passed by reference. If the complete list is not returned, set the value to `true`. The default value is `false`. (**Data type**: boolean)
- **maxResults**. The value configured in AP_Config.xml for the maximum results to return from the data source. This parameter can be used to query the data source to restrict results. However, a query to restrict the results is not mandatory as the abstraction layer restricts the results based on this value. (**Data type**: number)

The function returns an array of objects with the following properties.

- **Code**. Code for payment term (**Data type**: string; **Requirement**: Mandatory)
- **CalcMethod**. Calculation method of the payment term (**Data type**: string; **Requirement**: Optional)
- **Days**. Number of days for the payment, required to calculate invoice due date (**Data type**: string; **Requirement**: Optional)

Code syntax

```
CustomConnector.prototype.getPaymentTermsCodes = function(&moreResults, maxResults)
```

getCountryCurrencyRels

This function returns an array of all country-currency relationships to the AP Invoice eForm through the abstraction layer. The function is invoked when you run AP_Maintenance.js script or when the cache expires and the returned list is cached in AP_GetCompaniesCache.js file.

The following input parameters are passed to the function.

- **moreResults**. An indicator to denote whether more results are available or not. This value is passed by reference. If the complete list is not returned, set the value to `true`. The default value is `false`. (**Data type**: boolean)
- **maxResults**. The value configured in AP_Config.xml for the maximum results to return from the data source. This parameter can be used to query the data source to restrict results. However, a query to restrict the results is not mandatory as the abstraction layer restricts the results based on this value. (**Data type**: number)

The function returns an array of objects with the following properties.

- **CurrencyCode**. Currency code for the country (**Data type**: string; **Requirement**: Mandatory)
- **CountryCode**. Country code (**Data type**: string; **Requirement**: Mandatory)
- **LocaleCode**. Locale code (**Data type**: string; **Requirement**: Optional)

Code syntax

```
CustomConnector.prototype.getCountryCurrencyRels = function(&moreResults, maxResults)
```

getAOCCodes

This function returns an array of all add-on costs to the AP Invoice eForm through the abstraction layer. The function is invoked when you run AP_Maintenance.js script or when the cache expires and the returned list is cached in AP_GetCompaniesCache.js file.

The following input parameters are passed to the function.

- **moreResults**. An indicator to denote whether more results are available or not. This value is passed by reference. If the complete list is not returned, set the value to `true`. The default value is `false`. (**Data type**: boolean)
- **maxResults**. The value configured in AP_Config.xml for the maximum results to return from the data source. This parameter can be used to query the data source to restrict results. However, a query to restrict the results is not mandatory as the abstraction layer restricts the results based on this value. (**Data type**: number)

The function returns an array of objects with the following parameters.

- **Code**. Add-on cost code (**Data type**: string; **Requirement**: Mandatory)
- **Description**. Description of the add-on cost (**Data type**: string; **Requirement**: Mandatory)
- **txtGLBusUnit**. GL business unit number for the add-on cost (**Data type**: string; **Requirement**: Mandatory)
- **txtAcctgUnit**. Accounting unit for the add-on cost (**Data type**: string; **Requirement**: Mandatory)

- **txtGLAcct**. GL account number for the add-on cost (**Data type**: string; **Requirement**: Mandatory)
- **txtSubAcct**. Sub account number for the add-on cost (**Data type**: string; **Requirement**: Mandatory)
- **txtGLAcctDesc**. GL account description for the add-on cost (**Data type**: string; **Requirement**: Mandatory)
- **txtGLAcct5** to **txtGLAcct8**. GL account number (**Data type**: string; **Requirement**: Mandatory)
- **txtGL1** to **txtGL24**. Refers to a GL code that can be a profit center, cost center, WBS element, operation unit, or other such codes (**Data type**: string; **Requirement**: Optional)
- **txtGL1desc** to **txtGL24desc**. Description of the GL codes. (**Data type**: string; **Requirement**: Optional)

Code syntax

```
CustomConnector.prototype.getAOCCodes = function(&moreResults, maxResults)
```

getCompanyDetails

This function returns an object of company details for the provided company code to the AP Invoice eForm through the abstraction layer.

The following input parameters are passed to the function.

- **companyCode**. The company code whose details are to be extracted. (**Data type**: string)

The function returns an object with the following properties.

- **CompanyCode**. Company code (**Data type**: string; **Requirement**: Mandatory)
- **CompanyName**. Company name (**Data type**: string; **Requirement**: Mandatory)
- **City**. City (**Data type**: string; **Requirement**: Optional)
- **LanguageCode**. Language code of the country to which the company belongs (**Data type**: string; **Requirement**: Optional)
- **CountryKey**. Country key of the company (**Data type**: string; **Requirement**: Optional)
- **CurrencyCode**. Currency code (**Data type**: string; **Requirement**: Optional)
- **Vat**. VAT code (**Data type**: string; **Requirement**: Optional)

Code syntax

```
CustomConnector.prototype.getCompanyDetails = function(companyCode)
```

getTaxCodes

This function returns an array of tax code objects for provided country key, language code, and company code to the AP Invoice eForm through the abstraction layer.

Note The getTaxCodes function is invoked when you select a company from the Company drop-down in the AP Invoice eForm. If you implement the getVATCodes function and it returns a non-empty array, you need not implement the getTaxCodes function as the return object array structure is same for both the functions.

The following input parameters are passed to the function.

- **countryKey**. Key value of the country for which the tax codes are fetched (**Data type**: string)
- **languageCode**. Code of the language in which the tax code descriptions will be fetched (**Data type**: string)
- **companyCode**. Code of the company for which tax codes are to be fetched (**Data type**: string)
- **moreResults**. An indicator to denote whether more results are available or not. This value is passed by reference. If the complete list is not returned, set the value to `true`. The default value is `false`. (**Data type**: boolean)
- **maxResults**. The value configured in AP_Config.xml for the maximum results to return from the data source. This parameter can be used to query the data source to restrict results. However, a query to restrict the results is not mandatory as the abstraction layer restricts the results based on this value. (**Data type**: number)

The function returns an array of objects with the following parameters.

- **Code**. VAT code (**Data type**: string; **Requirement**: Mandatory)
- **Jurisdiction**. Jurisdiction corresponding to the VAT code (**Data type**: string; **Requirement**: Optional)
- **Rate**. VAT rate percentage (**Data type**: string; **Requirement**: Optional)
- **Purpose**. Description of the VAT code (**Data type**: string; **Requirement**: Optional)

Code syntax

```
CustomConnector.prototype.getTaxCodes = function(countryKey, languageCode,
companyCode, &moreResults, maxResults)
```

getVendors

This function returns an array of vendor objects for the provided vendor group, vendor name, vendor Id, and company code to the AP Invoice eForm through the abstraction layer.

The following input parameters are passed to the function.

- **vendorGroup**. Filter criteria to fetch vendors (**Data type**: string)
- **vendorName**. Filter criteria to fetch vendors (**Data type**: string)
- **vendorId**. Filter criteria to fetch vendors (**Data type**: string)
- **companyCode**. Filter criteria to fetch vendors (**Data type**: companyCode)
- **moreResults**. An indicator to denote whether more results are available or not. This value is passed by reference. If the complete list is not returned, set the value to `true`. The default value is `false`. (**Data type**: boolean)
- **maxResults**. The value configured in AP_Config.xml for the maximum results to return from the data source. This parameter can be used to query the data source to restrict results. However, a query to restrict the results is not mandatory as the abstraction layer restricts the results based on this value. (**Data type**: number)

The function returns an array of objects with the following properties.

- **VendorNum**. Vendor ID of the vendor (**Data type**: string; **Requirement**: Mandatory)
- **VendorName**. Vendor name (**Data type**: string; **Requirement**: Optional)

- **VendorGroup**. Vendor group of the vendor (**Data type**: string; **Requirement**: Optional)
- **Addr1**. Vendor address line 1 (**Data type**: string; **Requirement**: Optional)
- **Addr2**. Vendor address line 2 (**Data type**: string; **Requirement**: Optional)
- **Addr3**. Vendor address line 3 (**Data type**: string; **Requirement**: Optional)
- **Addr4**. Vendor address line 4 (**Data type**: string; **Requirement**: Optional)
- **City**. City corresponding to vendor address (**Data type**: string; **Requirement**: Optional)
- **State**. State corresponding to vendor address (**Data type**: string; **Requirement**: Optional)
- **Zip**. Zip code corresponding to vendor address (**Data type**: string; **Requirement**: Optional)
- **CountryCode**. Country code corresponding to vendor address (**Data type**: string; **Requirement**: Optional)
- **RemitToCode**. RemitTo code of the vendor (**Data type**: string; **Requirement**: Optional)
- **ShortName**. Short name of the vendor (**Data type**: string; **Requirement**: Optional)
- **PaymentTermsCode**. Payment terms code of the vendor (**Data type**: string; **Requirement**: Optional)
- **VAT**. VAT code (**Data type**: string; **Requirement**: Optional)
- **Field1**. Other details of the vendor (**Data type**: string; **Requirement**: Optional)
- **Field2**. Other details of the vendor (**Data type**: string; **Requirement**: Optional)

Code syntax

```
CustomConnector.prototype.getVendors = function(vendorGroup, vendorName, vendorId,
companyCode, &moreResults, maxResults)
```

getPO

This function returns an array of purchase order objects for the provided company code and vendor Id to the AP Invoice eForm through the abstraction layer.

The following input parameters are passed to the function.

- **companyCode**. Company code is a filter criteria to fetch purchase orders (**Data type**: string)
- **vendorId**. Vendor ID is a filter criteria to fetch purchase orders (**Data type**: string)
- **moreResults**. An indicator to denote whether more results are available or not. This value is passed by reference. If the complete list is not returned, set the value to `true`. The default value is `false`. (**Data type**: boolean)
- **maxResults**. The value configured in AP_Config.xml for the maximum results to return from the data source. This parameter can be used to query the data source to restrict results. However, a query to restrict the results is not mandatory as the abstraction layer restricts the results based on this value. (**Data type**: number)

The function returns an array of objects with the following properties.

- **CompanyCode**. Company code associated with a purchase order (**Data type**: string; **Requirement**: Mandatory)
- **PONumber**. Purchase order number (**Data type**: string; **Requirement**: Mandatory)

- **VendorID**. Vendor ID associated with a purchase order (**Data type**: string; **Requirement**: Mandatory)
- **RemitTo**. RemitTo code of the vendor (**Data type**: string; **Requirement**: Optional)
- **POAmount**. Total amount of the purchase order (**Data type**: float; **Requirement**: Optional)
- **POCurrency**. Currency code associated with a purchase order (**Data type**: string; **Requirement**: Mandatory)
- **PODate**. Purchase order date in the format yyyy-mm-dd (**Data type**: date; **Requirement**: Optional)
- **PODateMatched**. PO date matched in the format yyyy-mm-dd (**Data type**: date; **Requirement**: Optional)
- **DateClosed**. Date on which PO was closed, in the format yyyy-mm-dd (**Data type**: date; **Requirement**: Optional)

Code syntax

```
CustomConnector.prototype.getPO = function(companyCode, vendorId, &moreResults,
maxResults)
```

getVendorLocations

This function returns an array of vendor location objects for the provided vendor Id to the AP Invoice eForm through the abstraction layer.

The following input parameters are passed to the function.

- **vendorId**. ID of the vendor for which locations are to be fetched (**Data type**: string)
- **moreResults**. An indicator to denote whether more results are available or not. This value is passed by reference. If the complete list is not returned, set the value to `true`. The default value is `false`. (**Data type**: boolean)
- **maxResults**. The value configured in AP_Config.xml for the maximum results to return from the data source. This parameter can be used to query the data source to restrict results. However, a query to restrict the results is not mandatory as the abstraction layer restricts the results based on this value. (**Data type**: number)

The function returns an array of objects with the following parameters.

- **VendorID**. Vendor ID (**Data type**: string; **Requirement**: Mandatory)
- **Code**. Vendor location code (**Data type**: string; **Requirement**: Mandatory)
- **Name**. Vendor location name (**Data type**: string; **Requirement**: Mandatory)

Code syntax

```
CustomConnector.prototype.getVendorLocations = function(vendorId, &moreResults,
maxResults)
```

getPODetails

This function returns an object containing Purchase order details for the provided company code, vendor group, PO numbers, and PO vendor Id to the AP Invoice eForm through the abstraction layer.

The following input parameters are passed to the function.

- **companyCode**. Company code for which purchase order details are to be fetched (**Data type**: string)
- **vendorGroup**. Vendor group in which the company exists (**Data type**: string)
- **poNumbers**. Array of multiple PO numbers for which details are to be fetched (**Data type**: Array of string)
- **poVendorId**. Vendor ID of the vendor for the purchase order (**Data type**: string)
- **moreResults**. An indicator to denote whether more results are available or not. This value is passed by reference. If the complete list is not returned, set the value to `true`. The default value is `false`. (**Data type**: boolean)
- **maxResults**. The value configured in AP_Config.xml for the maximum results to return from the data source. This parameter can be used to query the data source to restrict results. However, a query to restrict the results is not mandatory as the abstraction layer restricts the results based on this value. (**Data type**: number)

The function returns an object that contains purchase order details with following properties.

- **POAmount**. Total amount of the purchase order (**Data type**: float; **Requirement**: Mandatory)
- **POCurrency**. Currency code associated with the purchase order amount (**Data type**: string; **Requirement**: Mandatory)
- **BlanketPOPresent**. Indicator to notify the presence of blanket purchase order (**Data type**: boolean; **Requirement**: Optional)
- **aPOList**. Array of all valid purchase order numbers (**Data type**: Array of string; **Requirement**: Mandatory)
- **rPOList**. Array of objects that contains properties for invalid PO number and error message
  - **PONumber**. Invalid PO number (**Data type**: string; **Requirement**: Mandatory)
  - **errorMsg**. Error message. For example, ERR_PO_NUM_NOT_FOUND, ERR_PO_CLOSED, ERR_VENDOR_DIFFERENT (**Data type**: string; **Requirement**: Mandatory)
- **VendorList**. Array of RemitTo vendor objects that contain same properties as in the return array of getVendors() (**Requirement**: Mandatory)
- **CodingLines**. Array of purchase order line item object with the following properties. (**Requirement**: Optional)
  - **BusUnitId**. Company code for the PO (**Data type**: string; **Requirement**: Mandatory)
  - **LinePONumber**. PO number for the line item (**Data type**: string; **Requirement**: Mandatory)
  - **LineNumber**. Line number (**Data type**: string; **Requirement**: Mandatory)
  - **ItemNumber**. Item number (**Data type**: string; **Requirement**: Mandatory)
  - **ItemDescription**. Description of the line item (**Data type**: string; **Requirement**: Optional)
  - **Quantity**. Quantity of the line item (**Data type**: float; **Requirement**: Mandatory)

- **OpenIVQuantityPO**. Invoice quantity available for the line item (**Data type**: float; **Requirement**: Mandatory)
- **UOM**. Unit of measure (**Data type**: string; **Requirement**: Mandatory)
- **UOMConvFactor**. Factor for conversion of order price unit into order unit. If the order price unit is not available, the value for this property must be set to `1`. (**Data type**: float; **Requirement**: Mandatory)
- **POPriceQuantity**. Total price quantity of the line item (**Data type**: float; **Requirement**: Optional)
- **POPriceUOM**. Order price unit (**Data type**: string; **Requirement**: Optional)
- **PriceUnit**. Price unit. If value for this property is not available, set the value to `1`. (**Data type**: float; **Requirement**: Mandatory)
- **UnitPrice**. Unit price of the item (**Data type**: float; **Requirement**: Mandatory)
- **ExtendedAmount**. Total line amount which is equal to Quantity * UnitPrice * UOMConvFactor / PriceUnit (**Data type**: float; **Requirement**: Mandatory)
- **OpenAmount**. Total line amount available (**Data type**: float; **Requirement**: Mandatory)
- **POLineTaxable**. Indicates whether PO line is taxable or not based on `true` or `false` value (**Data type**: string; **Requirement**: Mandatory)
- **POLVATJurisdiction**. VAT jurisdiction code for the line item (**Data type**: string; **Requirement**: Optional)
- **POLVATCode**. VAT code for the line item (**Data type**: string; **Requirement**: Optional)
- **GRDocument**. Goods receipt document number (**Data type**: string; **Requirement**: Optional)
- **GRFiscalYear**. Goods receipt fiscal year (**Data type**: string; **Requirement**: Optional)
- **GRItem**. Goods receipt item (**Data type**: string; **Requirement**: Optional)
- **ItemCategory**. Item category of line item (**Data type**: string; **Requirement**: Optional)
- **IsBlanketPOItem**. Indicator to notify blanket PO item (**Data type**: boolean; **Requirement**: Optional) (**Data type**: string; **Requirement**: Optional)
- **UpdateHost**. Indicator for valid PO line. Value for this property can be set as empty string to indicate valid line (**Data type**: string; **Requirement**: Optional)
- **PO1** to **PO10**. Additional fields (**Data type**: string; **Requirement**: Optional)

- **AccountAssignmentLines**. Array of purchase order account assignment lines with the following properties. (**Requirement**: Optional)
  - **PONumber**. Purchase order number with which the account assignment line is associated (**Data type**: string; **Requirement**: Mandatory)
  - **POLineNumber**. Purchase order line number with which the account assignment line is associated (**Data type**: string; **Requirement**: Mandatory)
  - **ItemNumber**. Item number of purchase order line with which the account assignment line is associated (**Data type**: string; **Requirement**: Mandatory)
  - **SerialNo**. Accounting data serial number (**Data type**: string; **Requirement**: Optional)
  - **GLAccount**. GL account number (**Data type**: string; **Requirement**: Optional)

- **CostCenter**. Cost center (**Data type**: string; **Requirement**: Optional)
- **ProfitCenter**. Profit center (**Data type**: string; **Requirement**: Optional)
- **WbsElement**. Work breakdown structure element (**Data type**: string; **Requirement**: Optional)
- **TaxJurisdictionCode**. Tax jurisdiction code for the line item (**Data type**: string; **Requirement**: Optional)
- **TaxCode**. Tax code for the line item (**Data type**: string; **Requirement**: Optional)

Code syntax

```
CustomConnector.prototype.getPODetails = function(companyCode, vendorGroup, poNumbers,
poVendorId, &moreResults, maxResults)
```

getGLAcctCodes

This function returns an array of GL account code objects for the provided partial GL account description to the AP Invoice eForm through the abstraction layer.

The following input parameters are passed to the function.

- **glAcctDescription**. Partial GL account description for which GL accounts are to be fetched (**Data type**: string)
- **moreResults**. An indicator to denote whether more results are available or not. This value is passed by reference. If the complete list is not returned, set the value to `true`. The default value is `false`. (**Data type**: boolean)
- **maxResults**. The value configured in AP_Config.xml for the maximum results to return from the data source. This parameter can be used to query the data source to restrict results. However, a query to restrict the results is not mandatory as the abstraction layer restricts the results based on this value. (**Data type**: number)

The function returns an array of objects with the following properties.

- **CompanyNum**. Company number of the GL account (**Data type**: string; **Requirement**: Mandatory)
- **GLUnit**. GL unit code of the GL account (**Data type**: string; **Requirement**: Mandatory)
- **GLAcct**. GL account number (**Data type**: string; **Requirement**: Mandatory)
- **SubAcct**. Sub account number (**Data type**: string; **Requirement**: Mandatory)
- **GLAcctDesc**. GL account description (**Data type**: string; **Requirement**: Mandatory)
- **GLAcct5** to **GLAcct8**. GL account additional information (**Data type**: string; **Requirement**: Mandatory)

Code syntax

```
CustomConnector.prototype.getGLAcctCodes = function(glAcctDescription, &moreResults,
maxResults)
```

getGLCodeList

This function returns an array of GL code objects for the provided GL number, GL codes that match the partial code input, and company code to the AP Invoice eForm through the abstraction layer.

The following input parameters are passed to the function.

- **constraint1**. A user constraint on the basis of which GL codes are filtered. The developer needs to ensure that `ALL` value for this parameter allows all users and GL code objects are not filtered. (**Data type**: string)
- **constraint2**. A user constraint on the basis of which GL codes are filtered. The developer needs to ensure that `ALL` value for this parameter allows all users and GL code objects are not filtered. (**Data type**: string)
- **glNumber**. Number corresponding to a GL field, for example, `1` for field “GL1,” `2` for field “GL2.”
- **glCode**. GL code. If partial GL value is provided, the function returns all GL codes starting with that value
- **companyCode**. Company code for which GL codes are to be fetched
- **moreResults**. An indicator to denote whether more results are available or not. This value is passed by reference. If the complete list is not returned, set the value to `true`. The default value is `false`. (**Data type**: boolean)
- **maxResults**. The value configured in AP_Config.xml for the maximum results to return from the data source. This parameter can be used to query the data source to restrict results. However, a query to restrict the results is not mandatory as the abstraction layer restricts the results based on this value. (**Data type**: number)
- **searchByDesc**. This parameter indicates whether the search should be performed using only value or using both value and description (in such a case, **glCode** acts as both partial value and partial description). (**Data type**: boolean)

The function returns an array of objects with the following properties.

- **constraint1**. A user constraint for the GL code (**Data type**: string; **Requirement**: Mandatory)
- **constraint2**. A user constraint for the GL code (**Data type**: string; **Requirement**: Mandatory)
- **glNumber**. Number corresponding to the GL field (**Data type**: string; **Requirement**: Mandatory)
- **GLCode**. GL code (**Data type**: string; **Requirement**: Mandatory)
- **Desc**. GL code description (**Data type**: string; **Requirement**: Mandatory)

Code syntax

```
CustomConnector.prototype.getGLCodeList = function(constraint1, constraint2, glNumber,
glCode, companyCode, &moreResults, maxResults)
```

getGLCodeUserConstraints

This function returns an array of GL code user constraint objects for the provided username to the AP Invoice eForm through the abstraction layer.

The following input parameters are passed to the function.

- **username**. Username of the user updating the GL (**Data type**: string)

- **moreResults**. An indicator to denote whether more results are available or not. This value is passed by reference. If the complete list is not returned, set the value to `true`. The default value is `false`. (**Data type**: boolean)
- **maxResults**. The value configured in AP_Config.xml for the maximum results to return from the data source. This parameter can be used to query the data source to restrict results. However, a query to restrict the results is not mandatory as the abstraction layer restricts the results based on this value. (**Data type**: number)

The function returns an array of objects with the following prerequisites.

- **Username**. Username of the GL user (**Data type**: string; **Requirement**: Mandatory)
- **GLUserValue1**. One of the constraints of the GL user (**Data type**: string; **Requirement**: Mandatory)
- **GLUserValue2**. One of the constraints of the GL user (**Data type**: string; **Requirement**: Mandatory)

Code syntax

```
CustomConnector.prototype.getGLCodeUserConstraints = function(username, &moreResults,
maxResults)
```

getPOCreditData

This function returns a PO credit data object for the provided company code, vendor group and PO number to the AP Invoice eForm through the abstraction layer.

The following input parameters are passed to the function.

- **companyCode**. Company code for which PO credit data is to be fetched (**Data type**: string)
- **vendorGroup**. Vendor group of the company (**Data type**: string)
- **poNumber**. Purchase order number for which purchase order credit data is to be fetched (**Data type**: string)
- **moreResults**. An indicator to denote whether more results are available or not. This value is passed by reference. If the complete list is not returned, set the value to `true`. The default value is `false`. (**Data type**: boolean)
- **maxResults**. The value configured in AP_Config.xml for the maximum results to return from the data source. This parameter can be used to query the data source to restrict results. However, a query to restrict the results is not mandatory as the abstraction layer restricts the results based on this value. (**Data type**: number)

The function returns an object having properties similar to the return object of getPODetails.

Code syntax

```
CustomConnector.prototype.getPOCreditData = function(companyCode, vendorGroup,
poNumber, &moreResults, maxResults)
```

getVendorVATIDData

This function returns an array of Vendor VAT ID data objects for provided company code and vendor Id to the AP Invoice eForm through the abstraction layer.

The following input parameters are passed to the function.

- **companyCode**. Company code for which vendor-VAT-ID data objects are to be fetched (**Data type**: string)
- **vendorId**. Vendor Id for which vendor-VAT-ID data objects are to be fetched (**Data type**: string)
- **moreResults**. An indicator to denote whether more results are available or not. This value is passed by reference. If the complete list is not returned, set the value to `true`. The default value is `false`. (**Data type**: boolean)
- **maxResults**. The value configured in AP_Config.xml for the maximum results to return from the data source. This parameter can be used to query the data source to restrict results. However, a query to restrict the results is not mandatory as the abstraction layer restricts the results based on this value. (**Data type**: number)

The function returns an array of objects with the following properties.

- **BusUnit**. Company code for the vendor (**Data type**: string; **Requirement**: Mandatory)
- **VendorID**. Vendor ID (**Data type**: string; **Requirement**: Mandatory)
- **VendorVATID**. VAT ID of the vendor (**Data type**: string; **Requirement**: Mandatory)

Code syntax

```
CustomConnector.prototype.getVendorVATIDData = function(companyCode, vendorId,
&moreResults, maxResults)
```

getSoldToVATIDData

This function returns an array of Sold-To-VAT-ID data objects for the provided company code to the AP Invoice eForm through the abstraction layer.

The following input parameters are passed to the function.

- **companyCode**. Company code for which Sold-To-VAT-Id objects are to be fetched (**Data type**: string)
- **moreResults**. An indicator to denote whether more results are available or not. This value is passed by reference. If the complete list is not returned, set the value to `true`. The default value is `false`. (**Data type**: boolean)
- **maxResults**. The value configured in AP_Config.xml for the maximum results to return from the data source. This parameter can be used to query the data source to restrict results. However, a query to restrict the results is not mandatory as the abstraction layer restricts the results based on this value. (**Data type**: number)

The function returns an array of objects with the following properties.

- **BusUnit**. Company code for the Sold-To-VAT-ID data object (**Data type**: string; **Requirement**: Mandatory)
- **SoldToVATID**. VAT ID of the vendor to which items are sold (**Data type**: string; **Requirement**: Mandatory)

Code syntax

```
CustomConnector.prototype.getSoldToVATIDData = function(companyCode, &moreResults,
maxResults)
```

getGLAcctDescription

This function returns GL account description for the provided GL account object to the AP Invoice eForm through the abstraction layer.

The following input parameters are passed to the function.

- **CompanyNum**. Company number of the GL account (**Data type**: string)
- **GLUnit**. GL unit code of the GL account (**Data type**: string)
- **SubAcct**. Sub account number (**Data type**: string)
- **GLAcct5** to **GLAcct8**. GL account additional information (**Data type**: string)
- GLAcctRowValue. GL account row value contains all the above GL account information separated by the delimiter "-". For example, CompanyNum-GLUnit-GLAcct-SubAcct-GLAcct5-GLAcct6-GLAcct7-GLAcct8. (**Data type**: string)

The function returns GL account description for the provided GL account object and empty string for an invalid GL account object.

Code syntax

```
CustomConnector.prototype.getGLAcctDescription = function(glAcctRow)
```

getHeaderLookupDataSet

This function returns an array of header lookup data for a source field (valid source field values: HeaderLookup1 to HeaderLookup10).

The following input parameters are passed to the function.

- **searchTerm**. Search term for which lookup results are desired. (**Data type**: string)
- **sourceName**. Source field name for which lookup results are desired - can be one of HeaderLookup1 to HeaderLookup10. (**Data type**: string)
- **filterParameterMap**. Filter parameter map object with parameters configured in AP_GetHeaderLookupDataSet's FilterParams section of AP_Config.xml based on which lookup results will be filtered (**Data type**: object)
- **moreResults**. An indicator to denote whether more results are available or not. This value is passed by reference. If the complete list is not returned, set the value to true. The default value is false. (**Data type**: boolean)
- **maxResults**. The value configured in AP_Config.xml for the maximum results to return from the data source. This parameter can be used to query the data source to restrict results. However, a query to restrict the results is not mandatory as the abstraction layer restricts the results based on this value. (**Data type**: number)

The function returns an array of header lookup data objects for a source field having the following properties:

- **value**. Value of the lookup data (**Data type**: string; Requirement: Mandatory)
- **desc**. Description of the lookup data (**Data type**: string; Requirement: Optional)

Code syntax

```
CustomConnector.prototype.getHeaderLookupDataSet = function(searchTerm, sourceName,
filterParameterMap, &moreResults, maxResults)
```