Vitalware Release Notes

ADO Reports

Vitalware 3.0

Document Version 1

Vitalware
Vital Records Management

AXIELL
REGISTRIES + VITAL RECORDS

Contents

SECTION 1

ADO Reports

Report generation and performance have been improved with Vitalware 3.0 and it is now possible to report directly to an Open Database Connectivity (ODBC) data source and to an ActiveX Data Objects (ADO) RecordSet object, bypassing the ODBC filtering process.

The new report options are:

- Crystal Reports: report directly in ODBC format, bypassing the ODBC filtering process.
- Crystal ADO: report using ADO RecordSets for Crystal (which are accessible via Crystal's ADO connector).
- Microsoft ADO: report using ADO RecordSets for Microsoft products.

> Crystal and Microsoft reports (Excel, Power Point and Word) which currently connect to an ODBC data source can be modified to use an ADO RecordSet.
> It remains possible to create reports by connecting directly to an ODBC data source.

Note

This document assumes familiarity with Report creation in Vitalware. Full details about Report Creation are available in the Vitalware Help: **Working with Vitalware records>Reports**.

SECTION 2

Crystal Reports

Creating a Crystal Report using the new ADO RecordSet is similar to creating a Crystal report with a direct ODBC connection. The main differences are in selecting the data source. This document describes the differences.

How to create a Crystal ADO Report

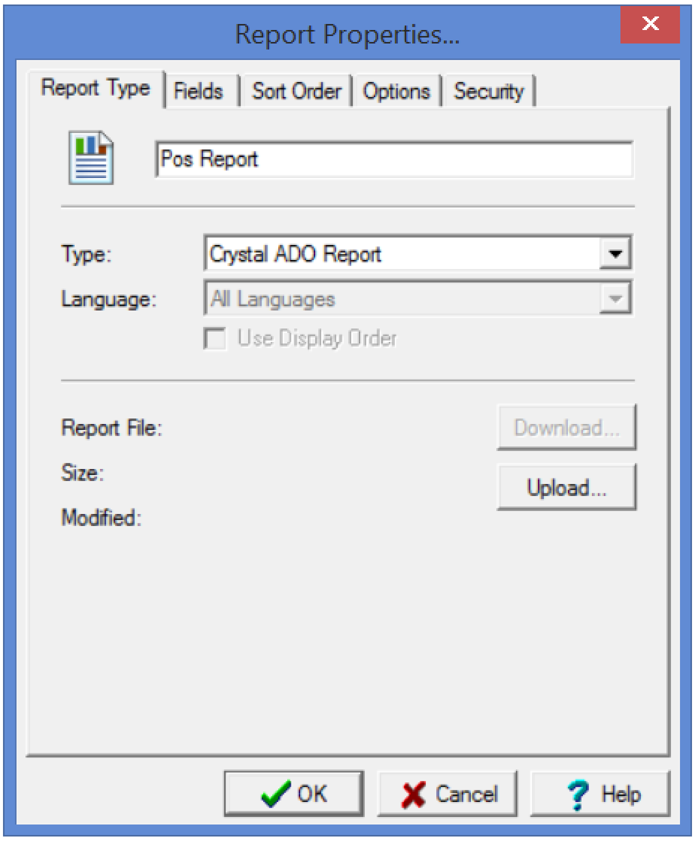
In Vitalware:

1. Search for or otherwise list a group of records on which to report.

> When designing a Crystal ADO report the records in your initial record set must have a value in each field to be included in the report. If not, the field name will not appear in the list of available columns. Once the report is defined, it does not matter if a record does not have values in every field included in the report.

2. Click **Reports** in the Tool bar to display the Reports box.
3. Click **New** in the Reports box.

   The Report Properties box displays.
4. Enter a descriptive name for the Report in the top text field.
5. Select Crystal ADO Report from the *Type* drop list:



Pos Report

Type: Crystal ADO Report

Language: All Languages

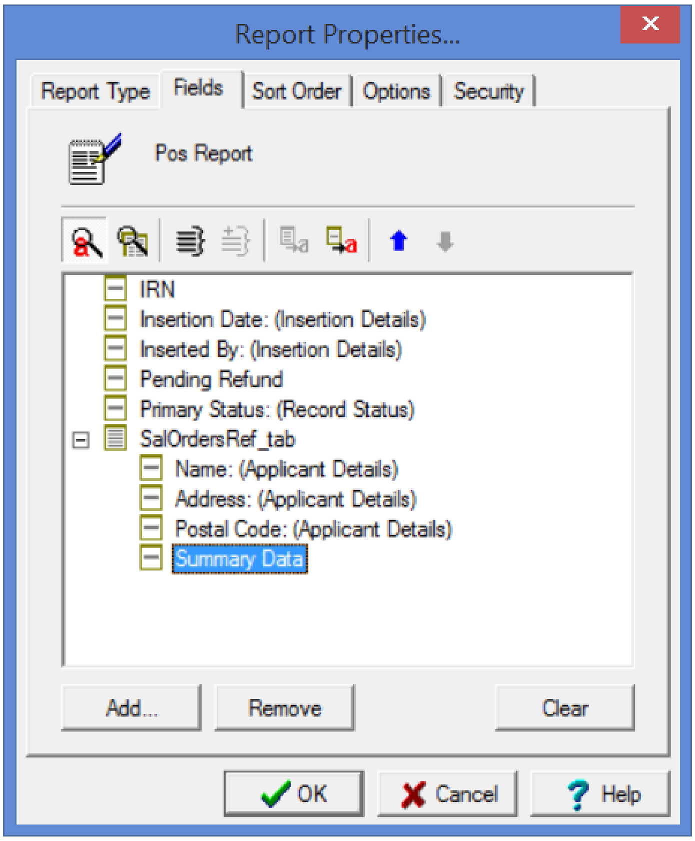
□ Use Display Order

Report File:

Size:

Modified:

6. On the Fields tab, add the fields to be included in the report. In this example the fields selected are:



Pos Report

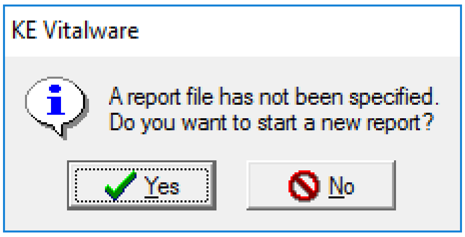
- IRN
- Insertion Date: (Insertion Details)
- Inserted By: (Insertion Details)
- Pending Refund
- Primary Status: (Record Status)
- SalOrdersRef_tab
  - Name: (Applicant Details)
  - Address: (Applicant Details)
  - Postal Code: (Applicant Details)
  - Summary Data

7. Make changes on the other tabs as required.

   See the Vitalware Help for details about setting a sort order, sort options, and security.
8. Click **OK**.

   The new report is added to the Reports box.
9. In the Reports box, select the new report and click to run the report for the first time.

   A message will display indicating that your report does not exist on the server. This is to be expected as the report has not yet been built in Crystal Reports:

   KE Vitalware

   A report file has not been specified. Do you want to start a new report?

10. Click **Yes**.

    An xml file is generated and saved with the data from your record set. The location of this file can vary, but typically it can be found in:

    `C:\Users\[`*your username*`]\AppData\Local\KESoftware\Reports\e[`*module name*`]`

    For example, a report run in the Parties module, will save the xmldata file to:

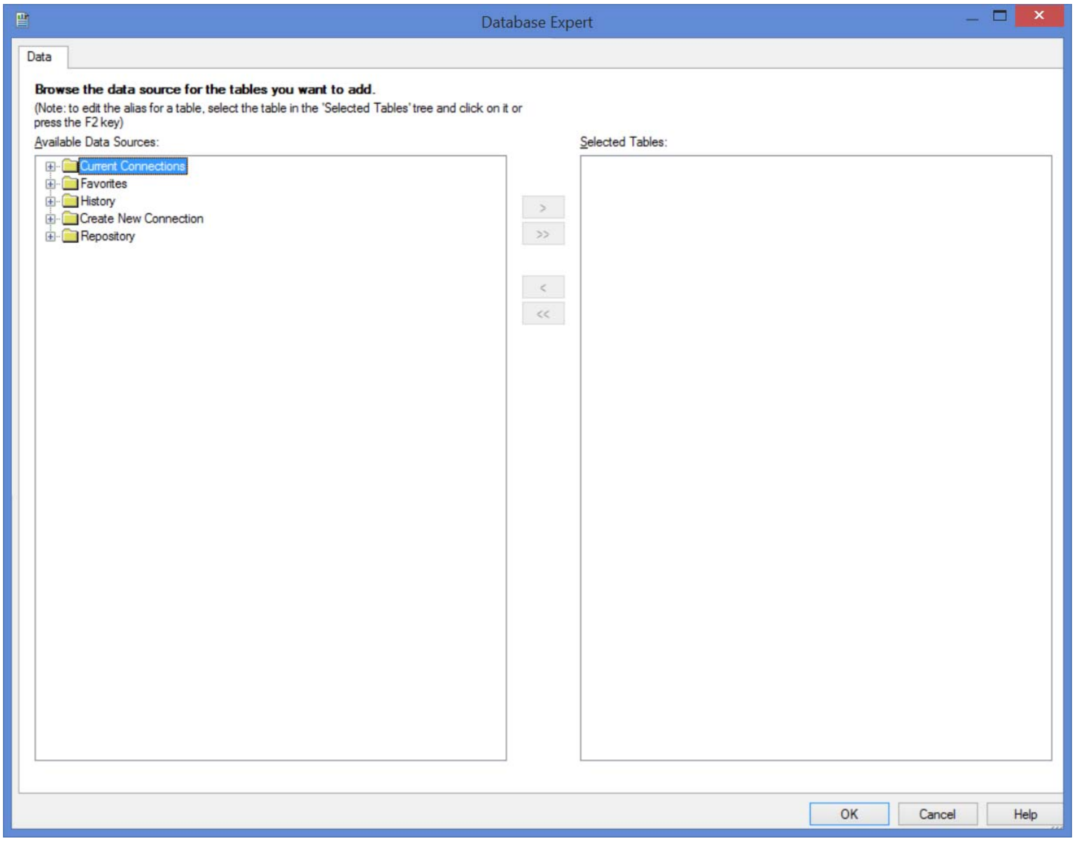
    `C:\Users\[`*your username*`]\AppData\Local\KESoftware\Reports\eparties`

    The Crystal Reports Designer application will open.
11. On the Start Page of the Crystal Reports Designer, select **Blank Report** under the New Reports heading

    -OR-

    Select **File>New>Blank Report** in the Menu bar.

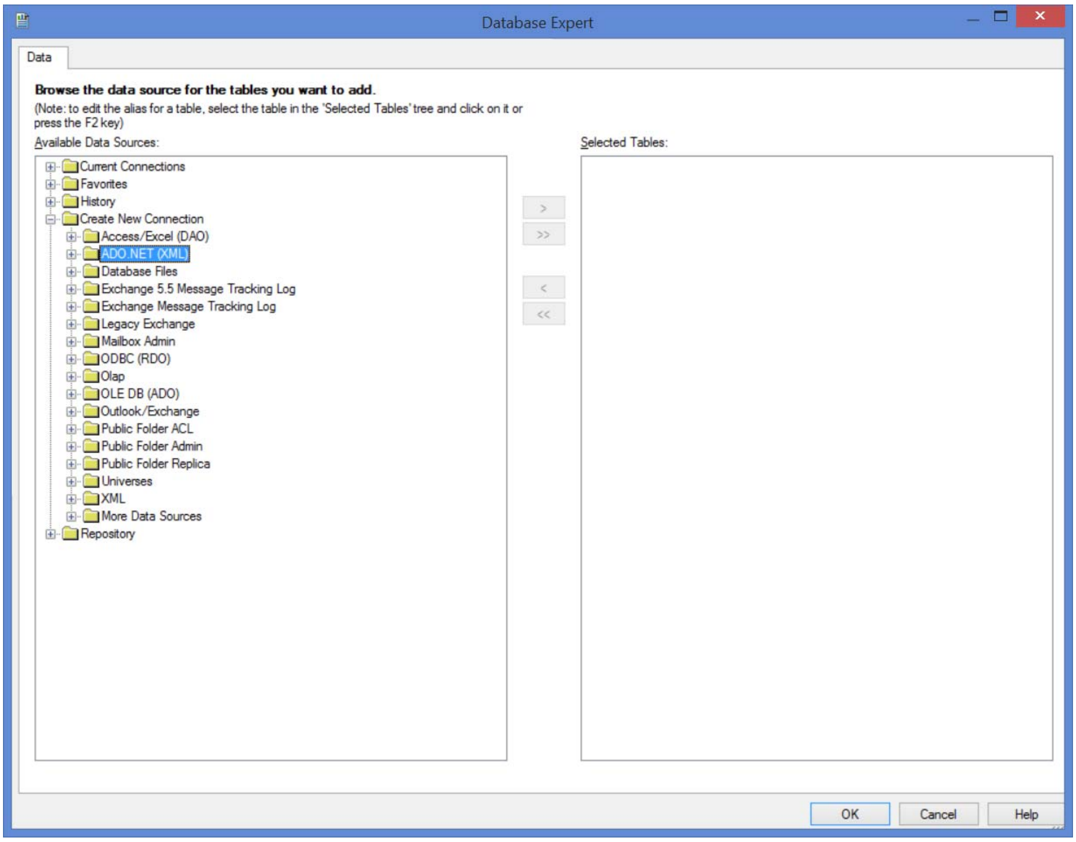
The Database Expert box displays:



Browse the data source for the tables you want to add.

(Note: to edit the alias for a table, select the table in the 'Selected Tables' tree and click on it or press the F2 key)

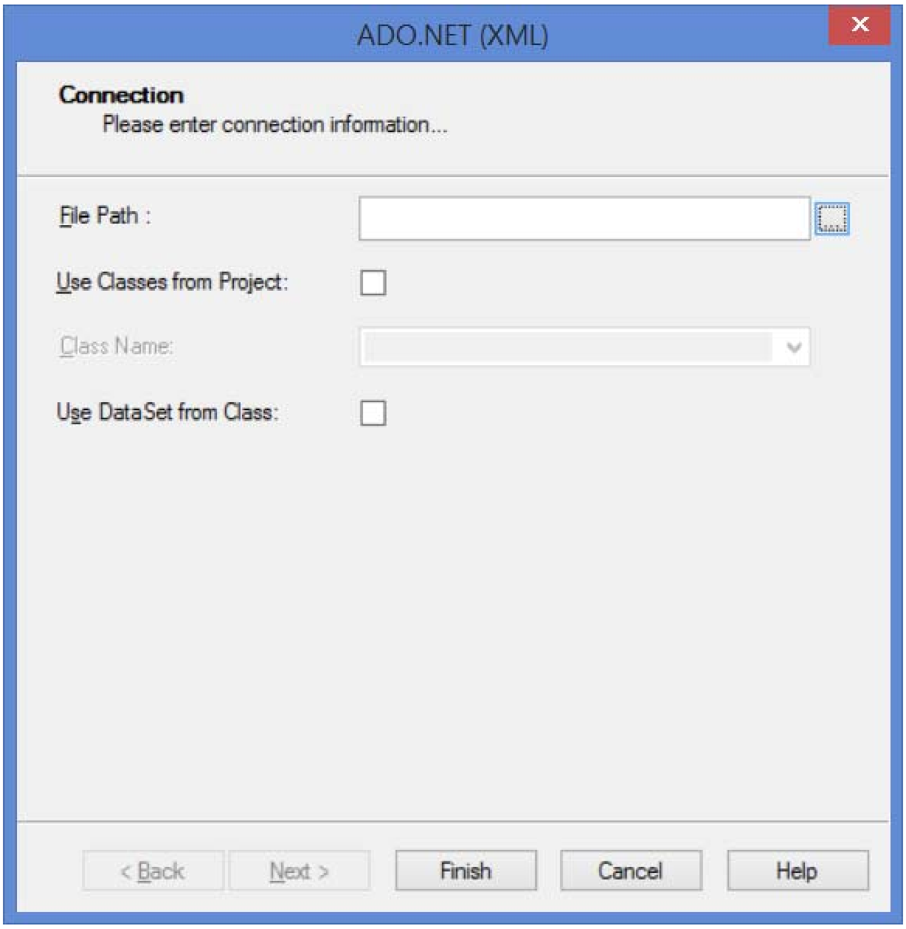
12. Double-click **Create New Connection** and click ⊞ beside **ADO.NET (XML)**:



Browse the data source for the tables you want to add.

(Note: to edit the alias for a table, select the table in the 'Selected Tables' tree and click on it or press the F2 key)

The following screen will display:

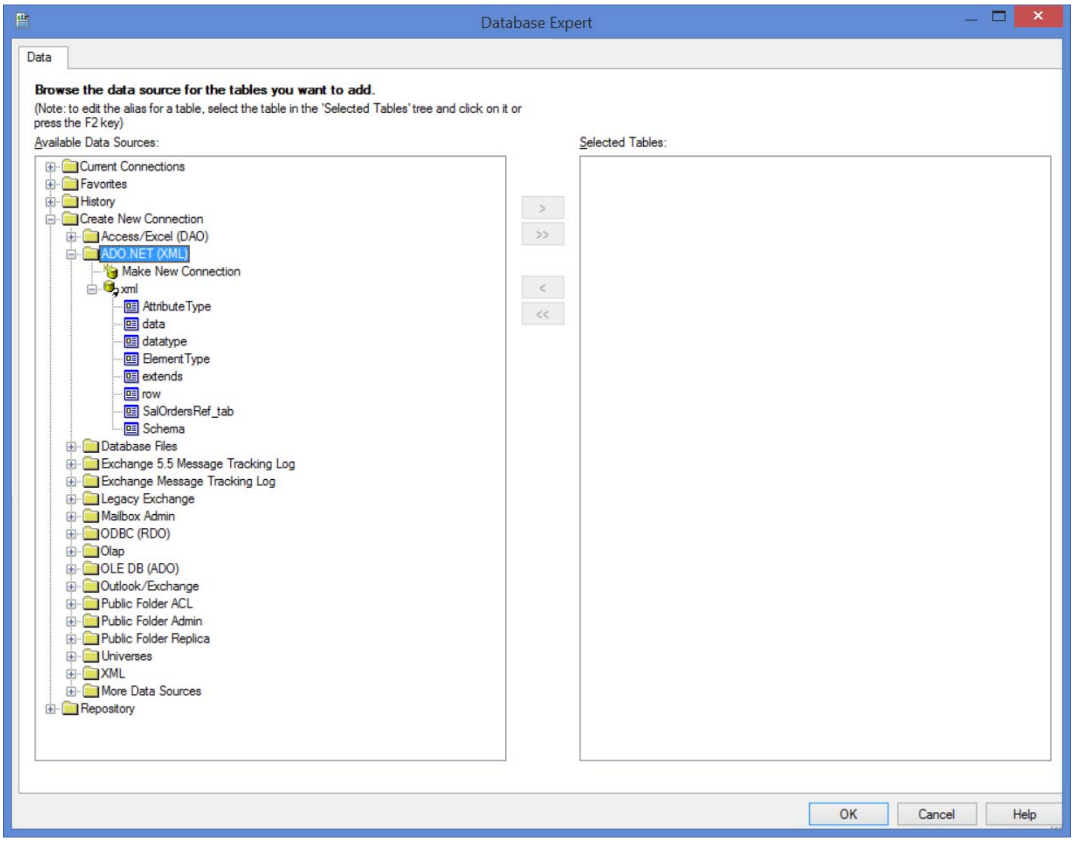


ADO.NET (XML)
Connection
Please enter connection information...
File Path :
Use Classes from Project: □
Class Name:
Use DataSet from Class: □
< Back
Next >
Finish
Cancel
Help

13. Click the button beside the *File Path* field to locate and select the `xmldata.xml` file created when the report was first run (Step 9).

    See Step 10 for details of the location of the `xmldata.xml` file.

14. Click **Finish** to return to the Database Expert:



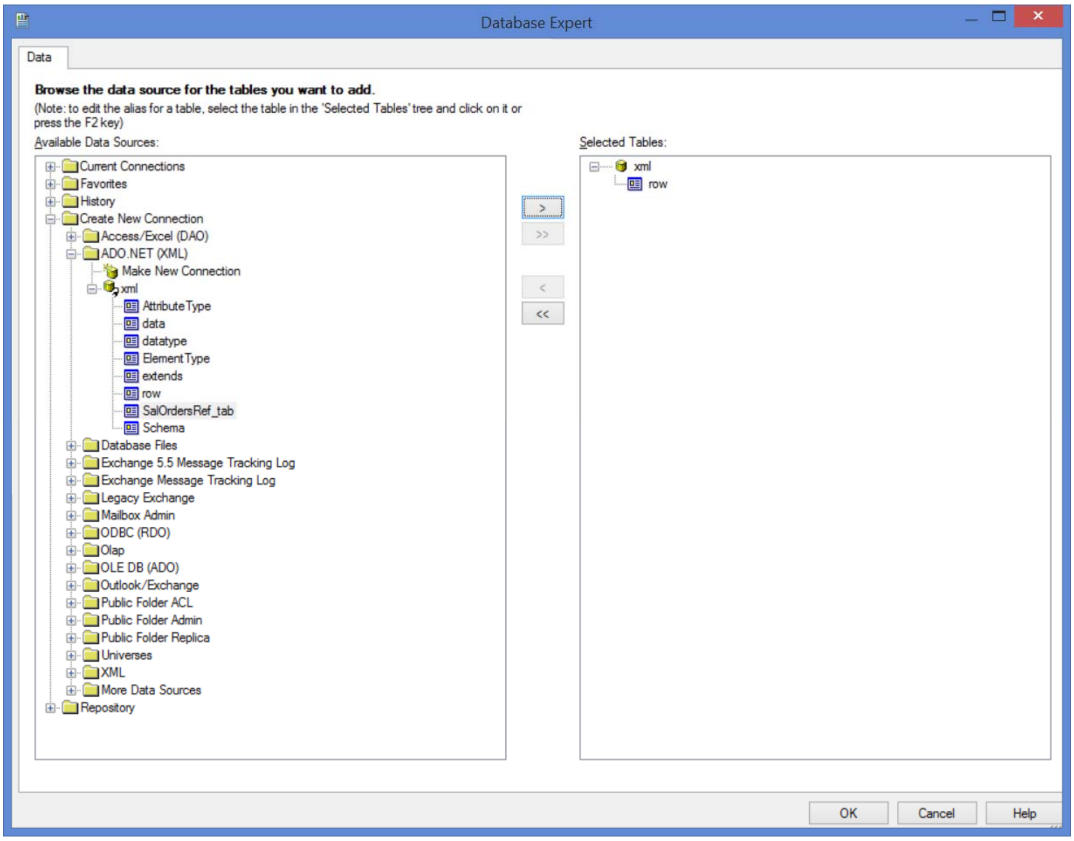
Database Expert

Browse the data source for the tables you want to add.

(Note: to edit the alias for a table, select the table in the 'Selected Tables' tree and click on it or press the F2 key)

Field values from the POS module are contained in the table called `row`.

15. Select **row** and add it to the *Selected Tables* pane:



Database Expert

Browse the data source for the tables you want to add.

(Note: to edit the alias for a table, select the table in the 'Selected Tables' tree and click on it or press the F2 key)

16. Click **OK**.

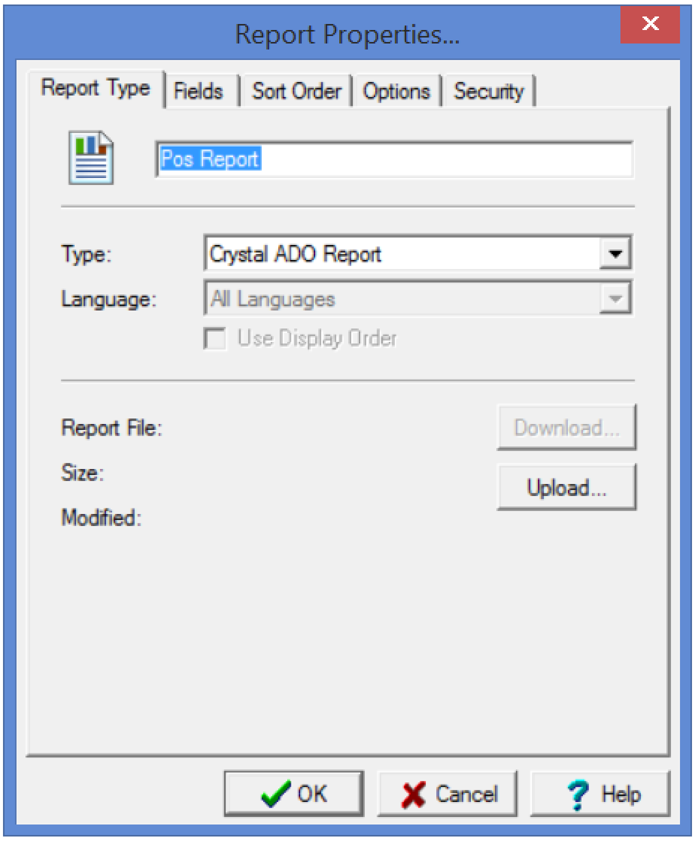
    The Crystal Report Designer displays, ready for you to design your Crystal report. See the Vitalware Help for details of designing a Crystal Report.

    > It is important not to move the `xmldata.xml` file as this will cause problems when sharing the report with other users.

How to modify a Crystal Report to use ADO instead of ODBC

To modify a Crystal Report to use ADO rather than ODBC:

1. Open the Report Properties dialog for the report.
2. Select **Crystal ADO Report** from the Type drop list:



Pos Report

Type: Crystal ADO Report

Language: All Languages

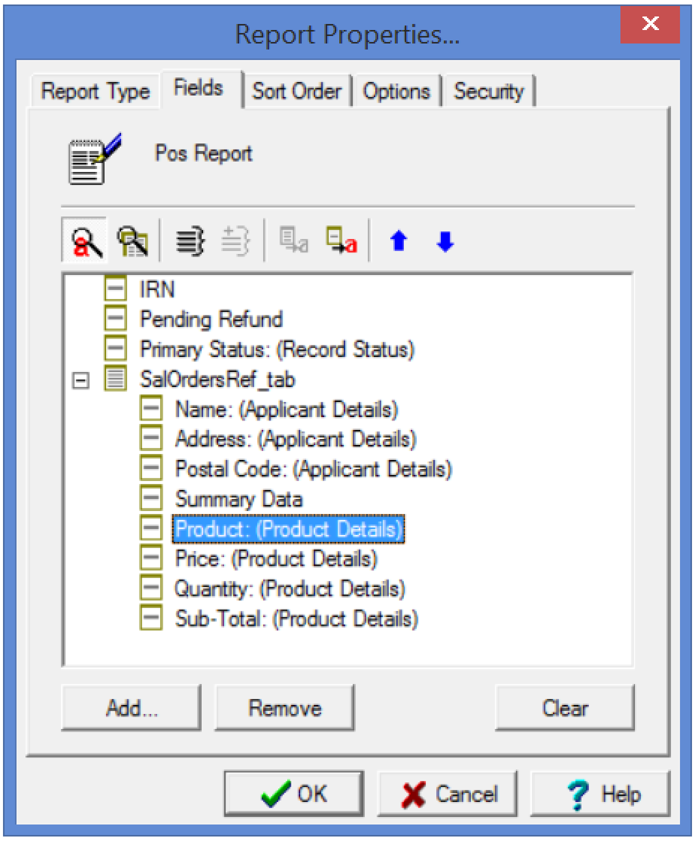
□ Use Display Order

Report File:

Size:

Modified:

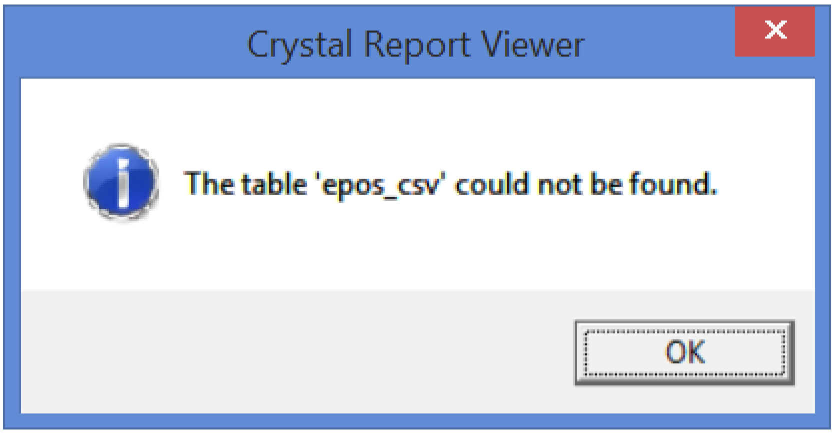
The fields for this report are:


Report Properties...
Report Type
Fields
Sort Order
Options
Security
Pos Report
IRN
Pending Refund
Primary Status: (Record Status)
SalOrdersRef_tab
Name: (Applicant Details)
Address: (Applicant Details)
Postal Code: (Applicant Details)
Summary Data
Product: (Product Details)
Price: (Product Details)
Quantity: (Product Details)
Sub-Total: (Product Details)
Add...
Remove
Clear
OK
Cancel
Help

Two tables are generated in this report.

3. Click **OK** and run the report.

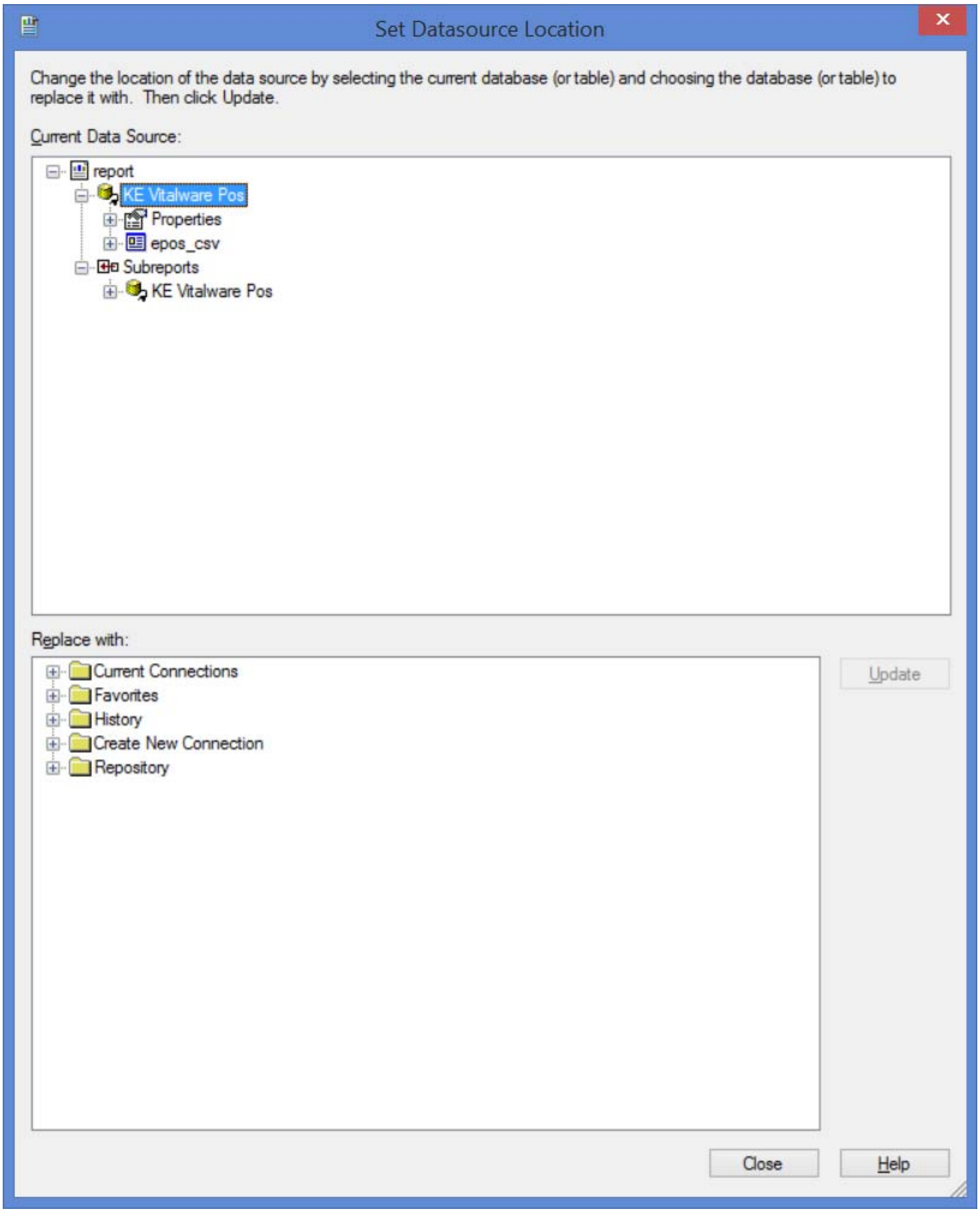
   Crystal will create the ADO record set and the following error will display:



The table 'epos_csv' could not be found.

4. Open the Crystal report in the Crystal Report Designer and select the **Database>Set Datasource Location** menu option.

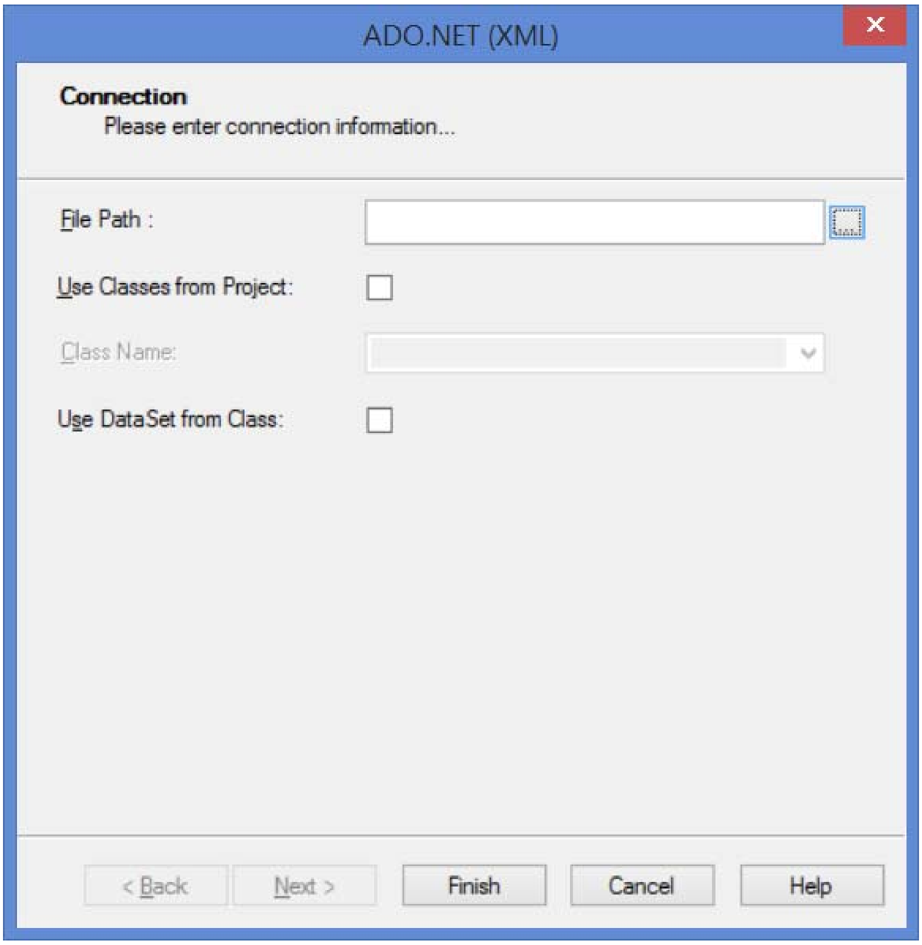
   The Set Datasource Location dialog will display:



Change the location of the data source by selecting the current database (or table) and choosing the database (or table) to replace it with. Then click Update.

5. Select **Create New Connection** in the *Replace with* pane and click ⊞ beside **ADO.NET (XML)**.

   The following screen will display:



Please enter connection information...

File Path :

Use Classes from Project: □

Use DataSet from Class: □

6. Click the button beside the *File Path* field to locate and select the `xmldata.xml` file created when the report was run.

   The location of this file can vary, but typically it can be found in:

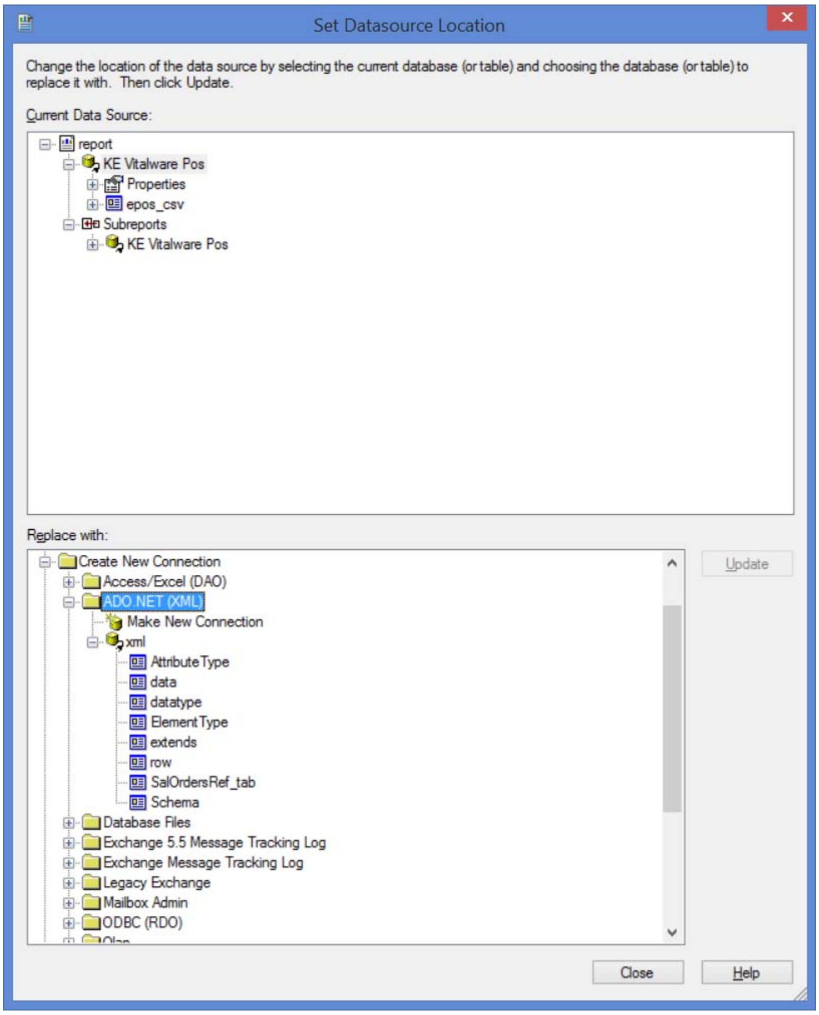
   C:\Users\[*your username*]\AppData\Local\KESoftware\Reports\e[*module name*]

   For example, a report run in the Parties module, will save the xmldata file to:

   C:\Users\[*your username*]\AppData\Local\KESoftware\Reports\eparties

7. Click **Finish.**

   You are returned to the Set Datasource Location dialog:



Change the location of the data source by selecting the current database (or table) and choosing the database (or table) to replace it with. Then click Update.

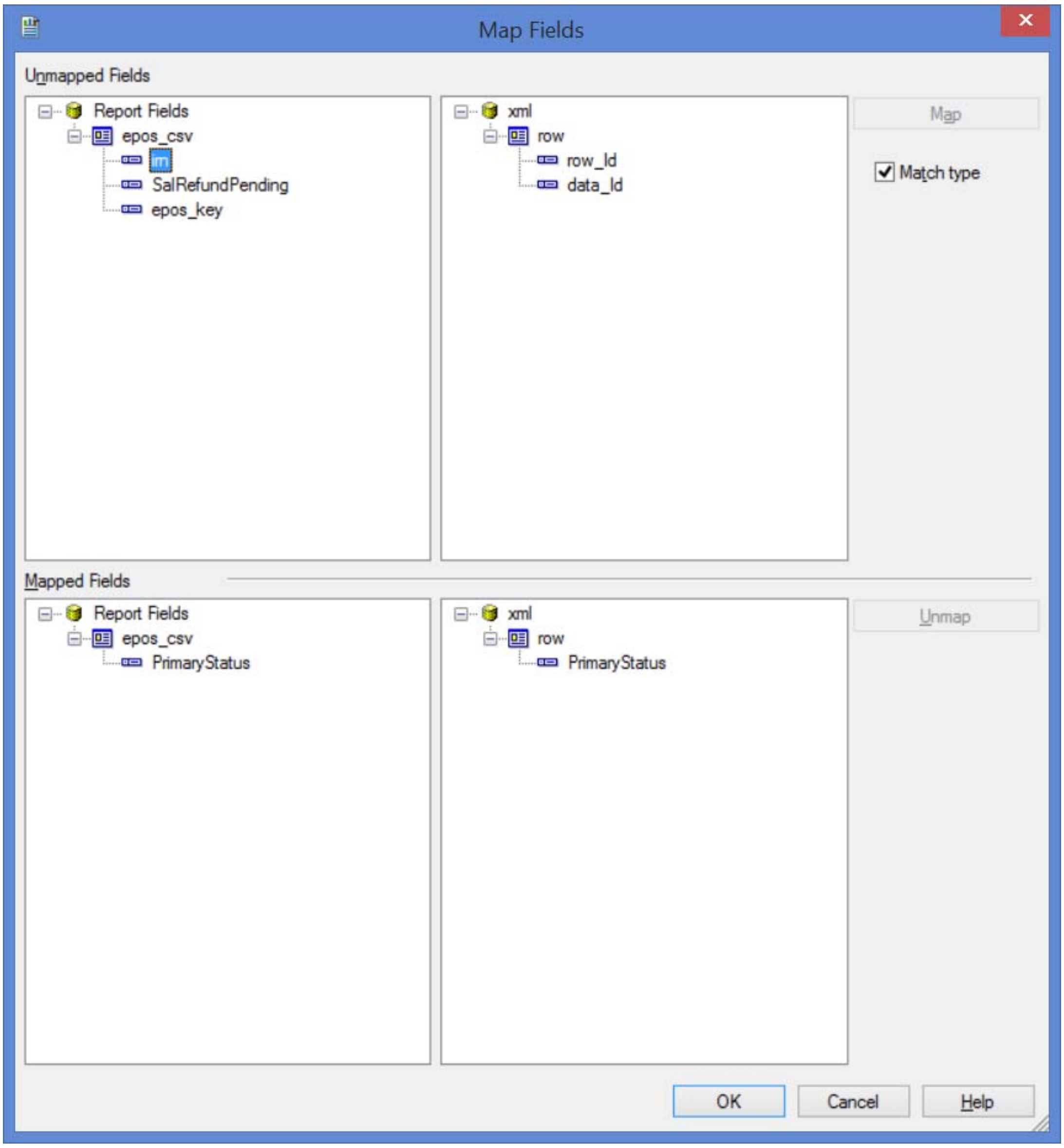
Next it is necessary to map fields from the old ODBC data source to the new ADO RecordSet.

In this example there are two tables to map and one sub-report.

8. To map the old ODBC POS fields to the new POS table, click **epos_csv** in the *Current Data Source* pane and then click the **row** table in the *Replace with* pane.

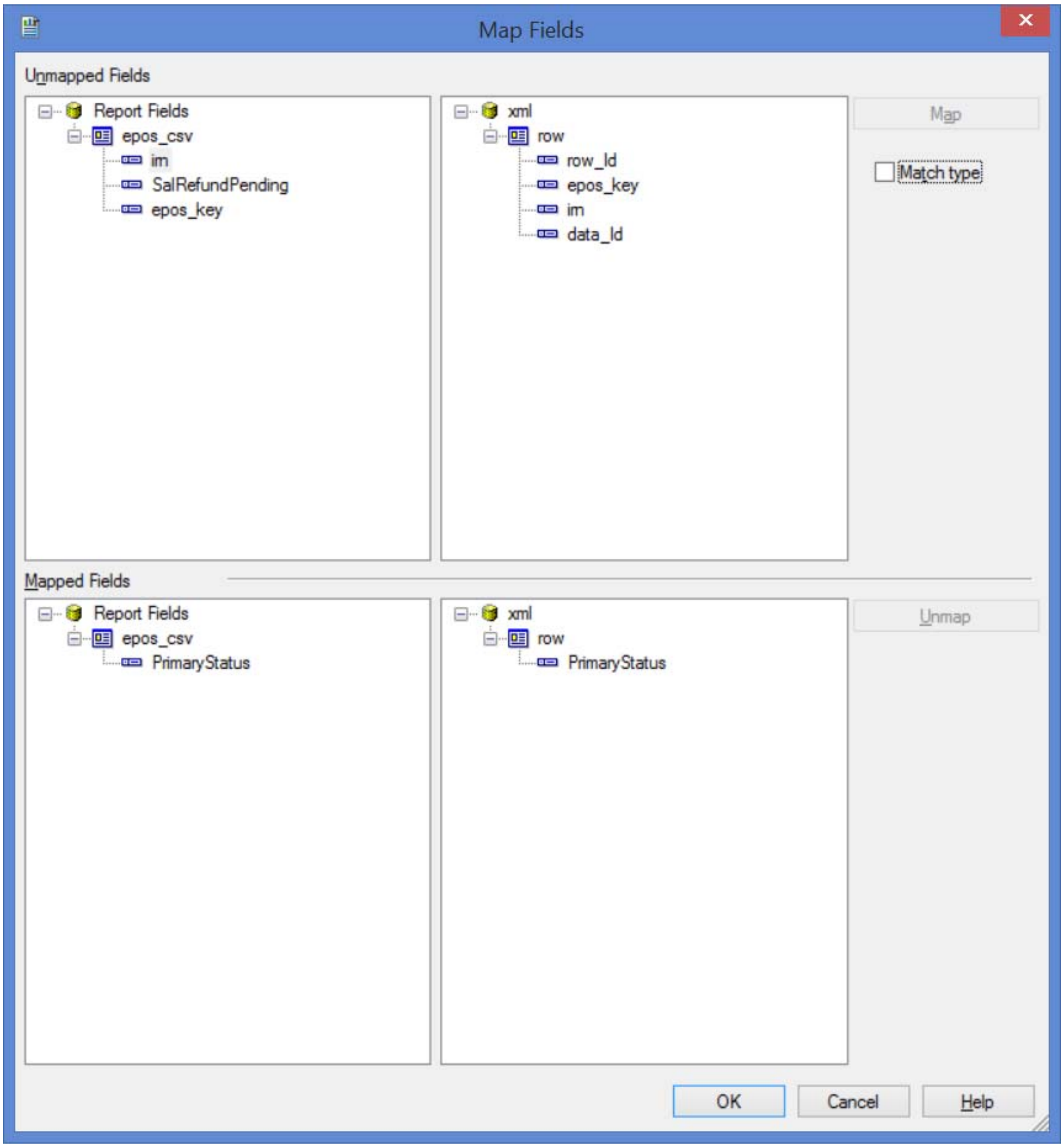
   The Update button will be enabled.

9. Click the **Update** button and the Map Fields dialog will display:



Fields with the same name will be mapped automatically.

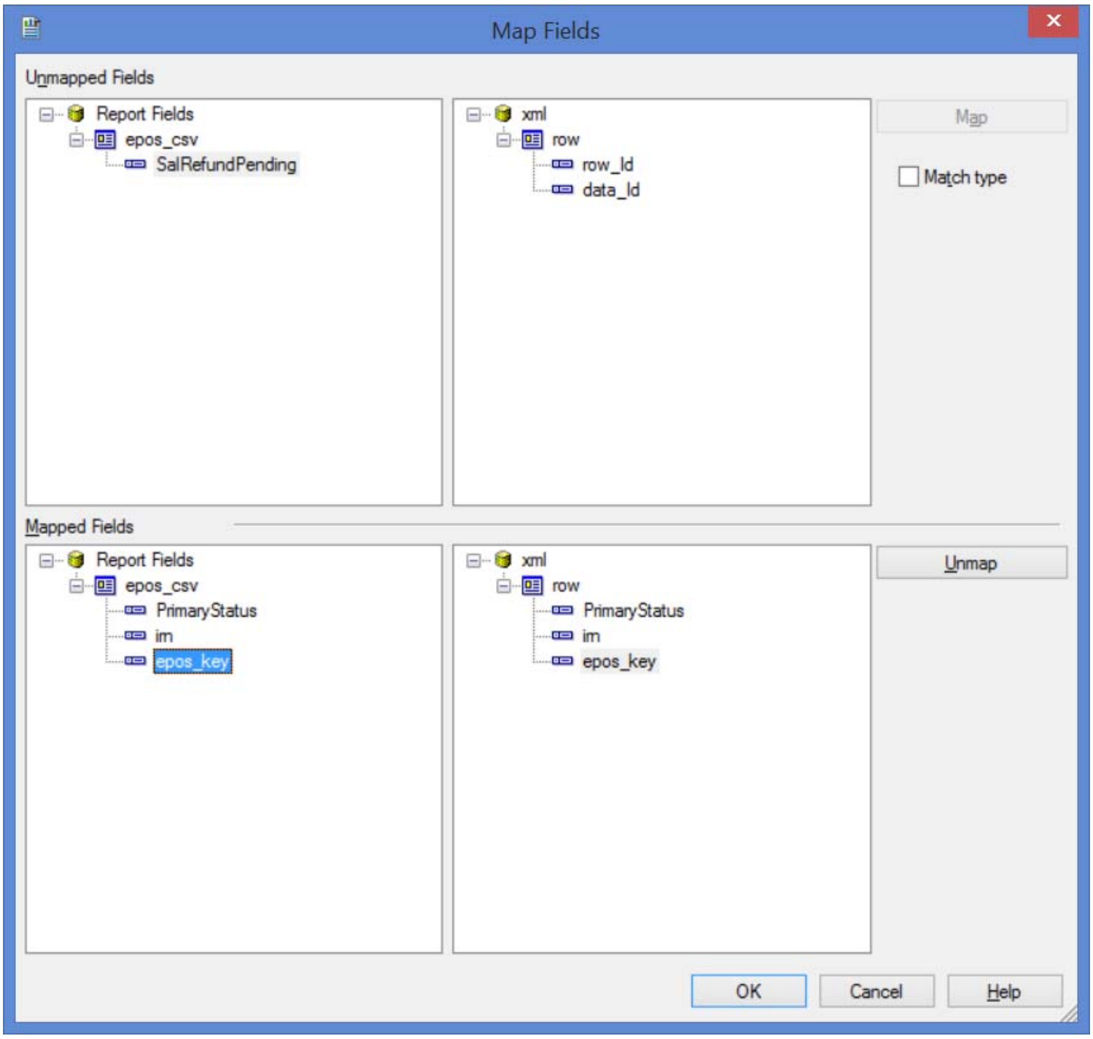
10. Uncheck the **Match type** check box to reveal more fields in the *Unmapped Fields* pane:



11. Complete mapping fields in the *Unmapped Fields* pane.

    In this example we map `epos_key` to `epos_key` and `irn` to `irn` by selecting both fields to map and clicking the **Map** button.

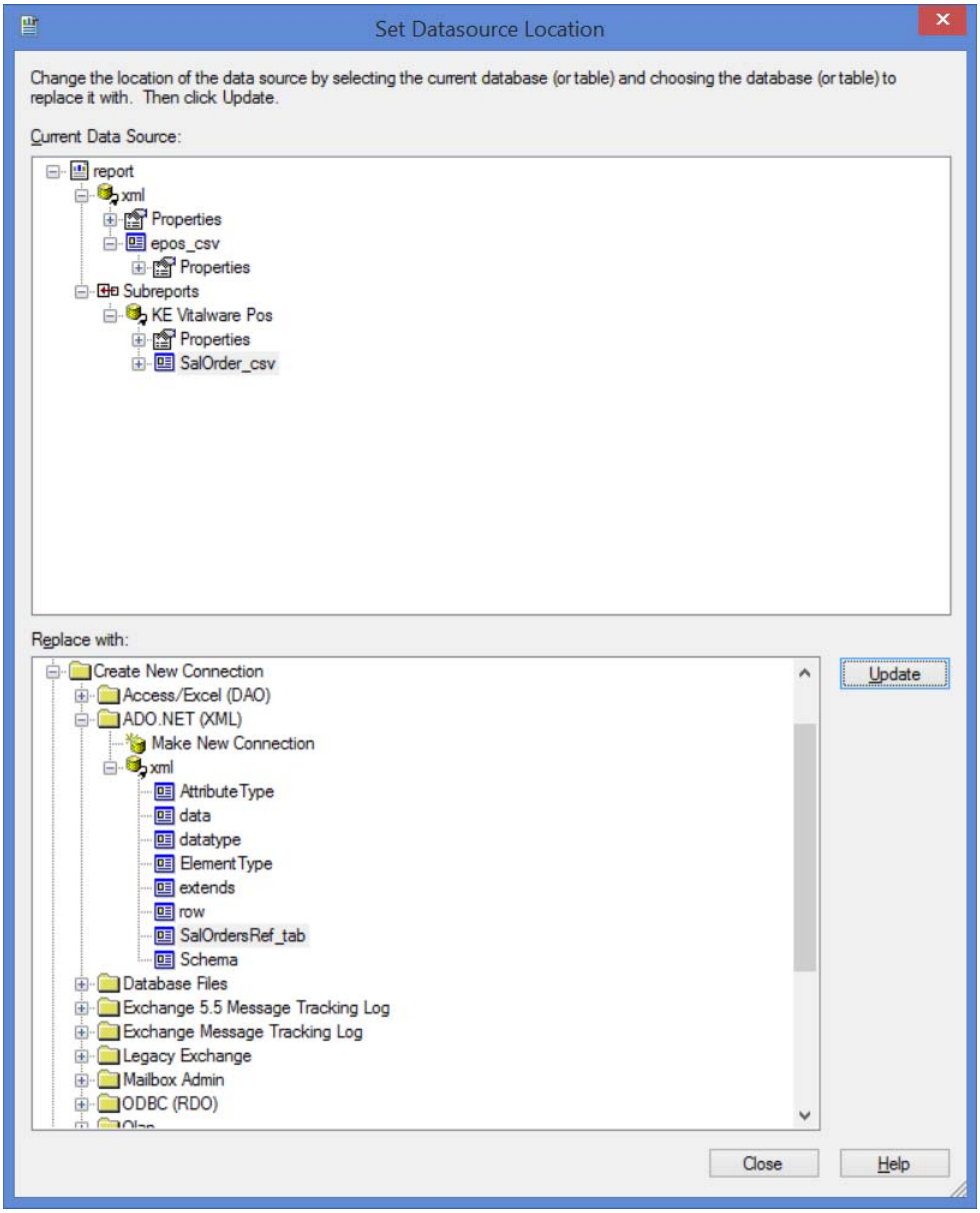
    Once mapped, fields will be moved to the *Mapped Fields* pane:



12. Click **OK** when all fields are mapped.

    You are returned to the Set Datasource Location dialog.

13. Repeat the mapping process for all fields (in this example, mapping fields in the `SalOrder_csv` table to the ADO table `SalOrdersRef_tab`):



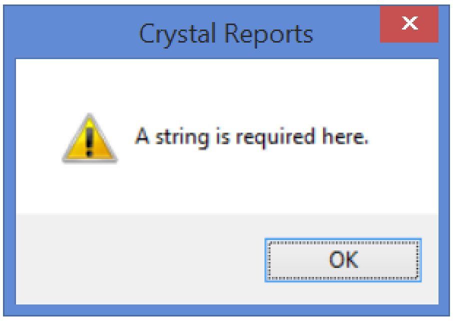
Set Datasource Location

Change the location of the data source by selecting the current database (or table) and choosing the database (or table) to replace it with. Then click Update.

14. Once all fields have been remapped in all tables click **Close**.

    You are returned to the Crystal design window.

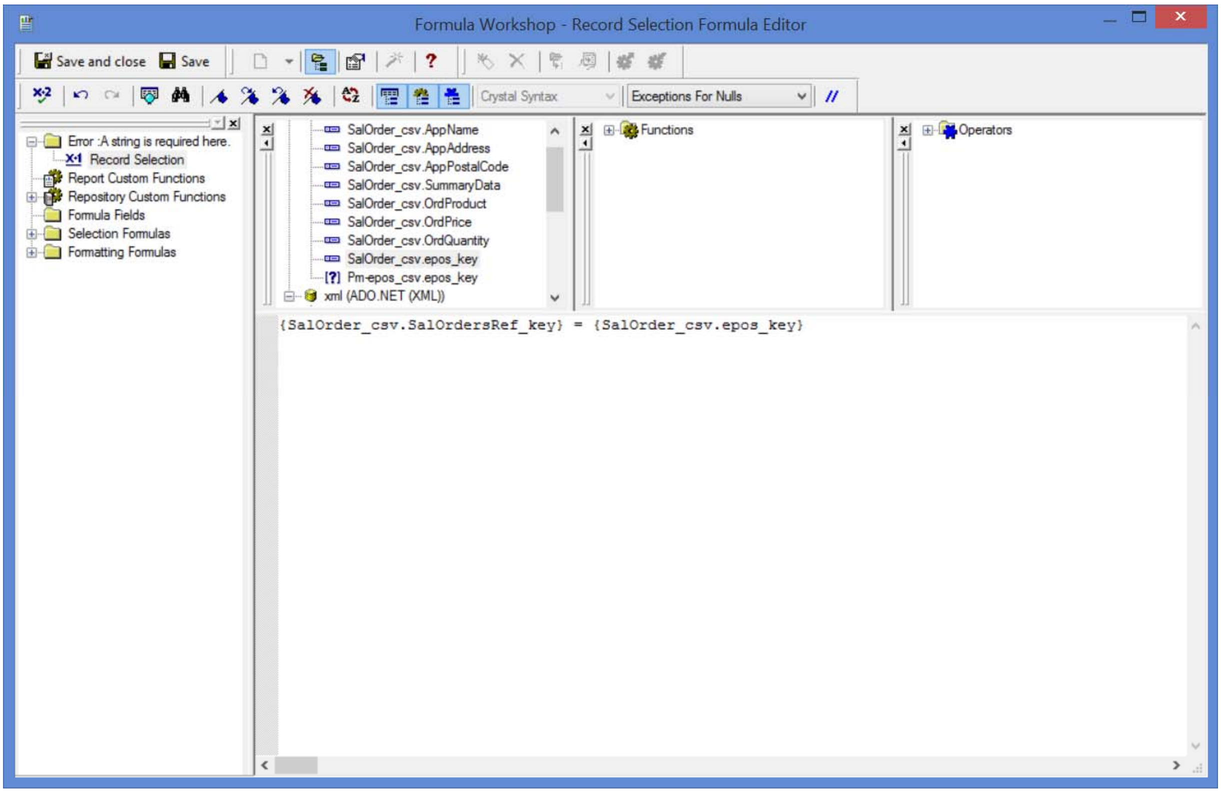
    If you refresh report data at this stage and you have a sub-report object, you will probably receive an error regarding sub-report links, e.g.:



Crystal Reports

A string is required here.

Click **OK** to open the Record Selection Formula Editor. Change the link key field used by the old ODBC table to the link key field referenced by the ADO RecordSet:



Formula Workshop - Record Selection Formula Editor

```
{SalOrder_csv.SalOrdersRef_key} = {SalOrder_csv.epos_key}
```

The report should now work correctly.

SECTION 3

Microsoft Excel

The following examples demonstrate how to create a basic Excel report using VBA. Please note that it is not the intention of this document to teach VBA.
Excel 2013 was used to create these reports.

How to create an Excel Report using the ADO RecordSet

With ODBC data sources there is an option in Excel to open a connection without writing Visual Basic code. This is not the case when making a connection to an ADO record set and it is necessary to write VB code.

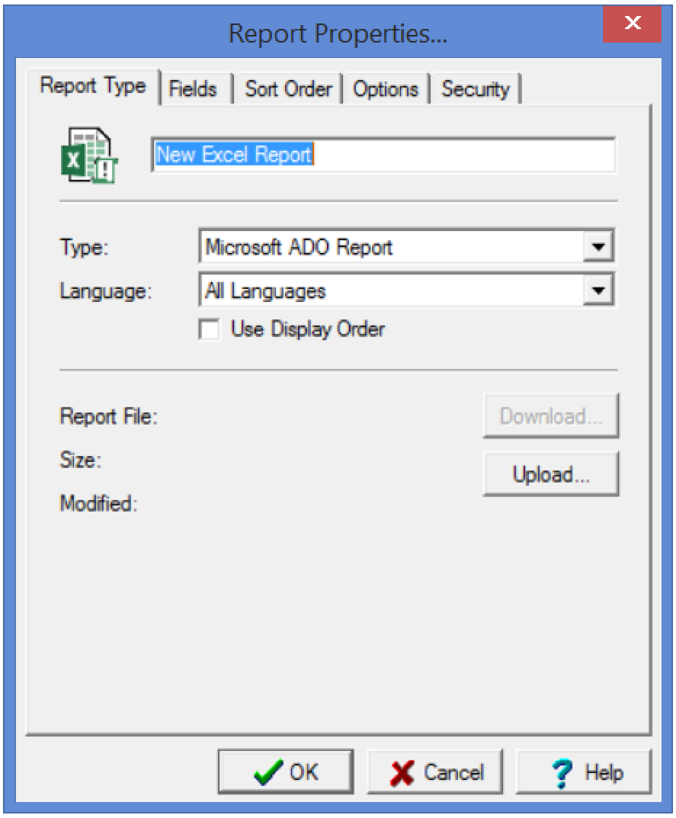
Step 1: Create a new report in Vitalware

This first example is a simple report on single value fields from the POS module. The VBA code provided in this example will automatically populate headings and row data for each column selected.

In Vitalware:

1. Search for or otherwise list a group of records on which to report.
2. Click **Reports** in the Tool bar to display the Reports box.
3. Click **New** in the Reports box.

   The Report Properties box displays.
4. Enter a descriptive name for the Report in the top text field.
5. Select Microsoft ADO Report from the *Type* drop list:



New Excel Report

Type: Microsoft ADO Report

Language: All Languages

□ Use Display Order

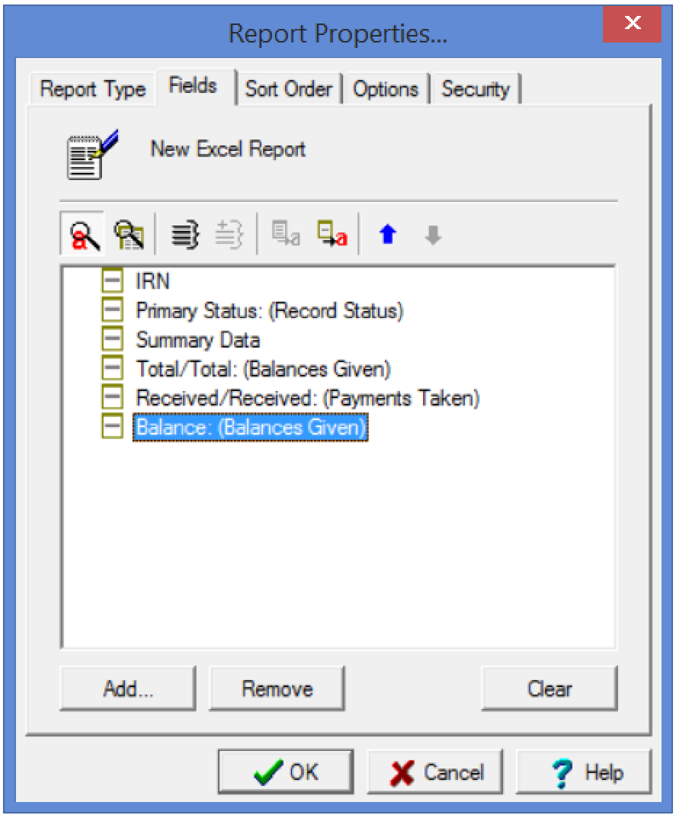
Report File:

Size:

Modified:

6. On the **Fields** tab add the fields to be included in the report.

   Fields selected in this example are:



New Excel Report

- IRN
- Primary Status: (Record Status)
- Summary Data
- Total/Total: (Balances Given)
- Received/Received: (Payments Taken)
- Balance: (Balances Given)

7. Make changes on the other tabs as required.

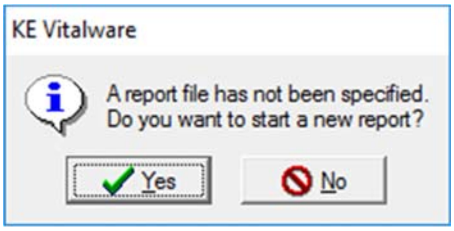
   See the Vitalware Help for details about setting a sort order, sort options, and security.

8. Click **OK**.

   The new report is added to the Reports dialog box.

9. Select the new report and click **Report All** to run the report for the first time.

   A message will display indicating that your report does not exist on the server. This is to be expected as the report has not yet been built in Excel:



A report file has not been specified.
Do you want to start a new report?

10. Click **Yes**.

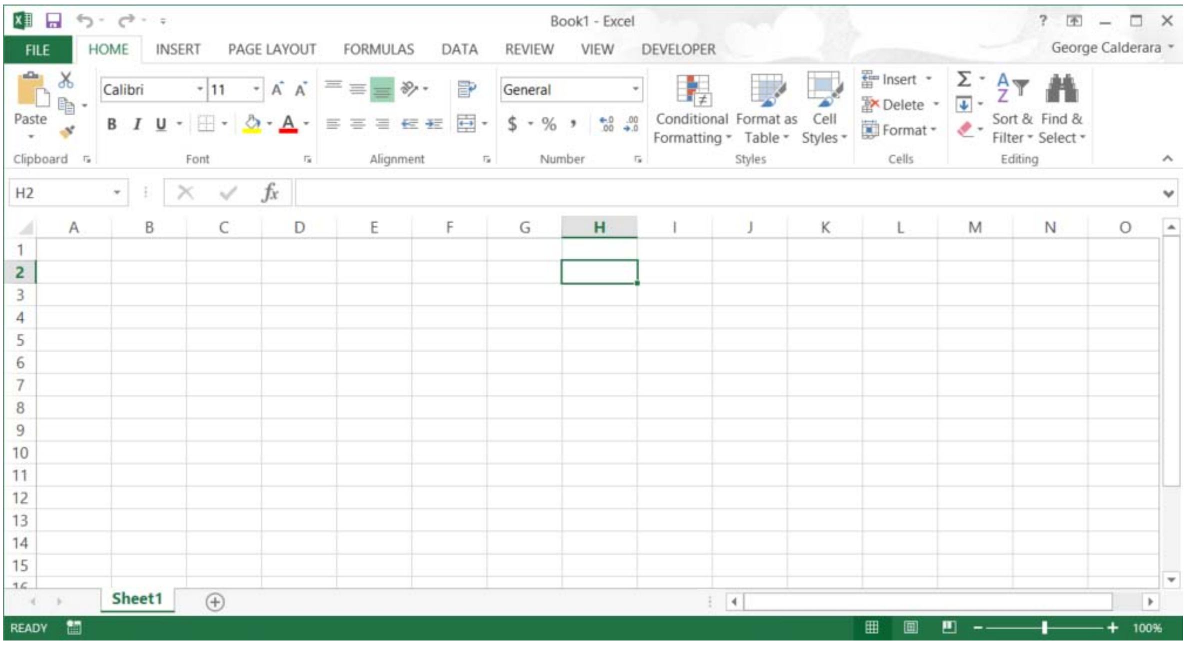
    An xml file is generated and saved with the data from your record set. The location of this file can vary, but typically it can be found in:

    `C:\Users\[`*your username*`]\AppData\Local\KESoftware\Reports\e[`*module name*`]`

    For example, a report run in the Parties module, will save the xmldata file to:

`C:\Users\[`*`your username`*`]\AppData\Local\KESoftware\Reports\eparties`

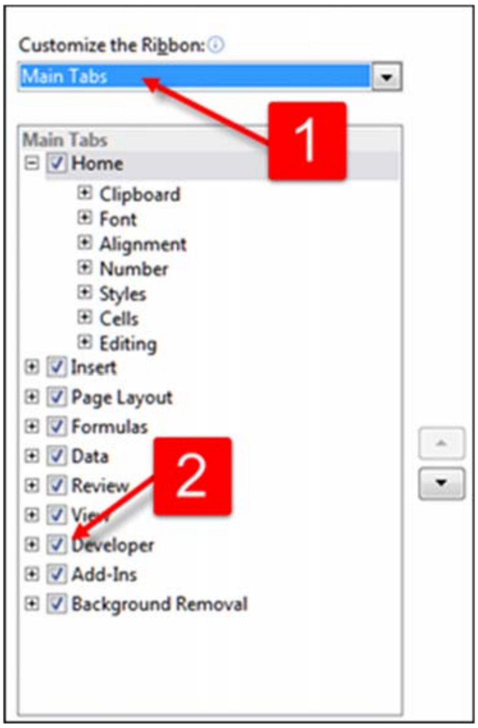
Microsoft Excel will open with a blank worksheet as follows:



Ensure that Excel is setup correctly

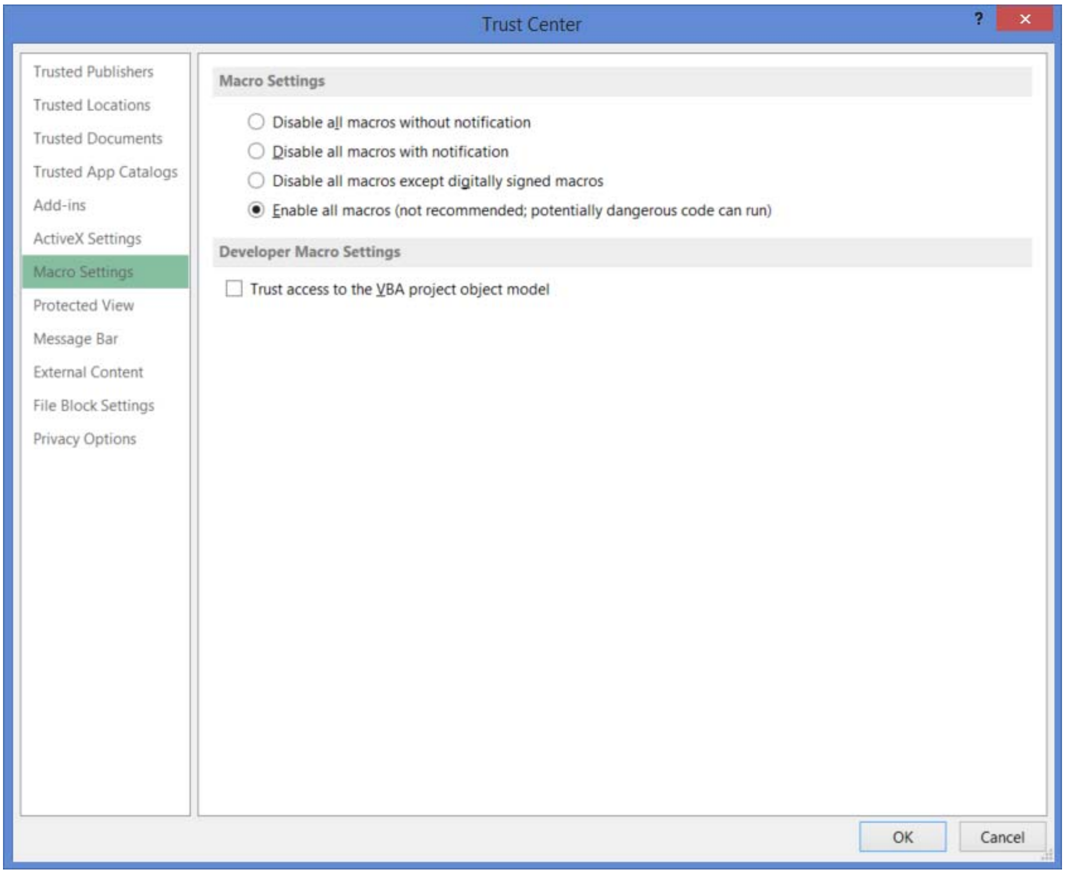
If the Developer tab does not display in the Ribbon:

1. Click **File>Options>Customize Ribbon**.
2. With **Main Tabs** selected from the *Customize the Ribbon* drop list (1), select the **Developer** check box (2):



In order to run the macros that we will create with our reports, we need to ensure that the Security level in Excel is appropriate:

1. On the Developer tab, click .
2. Enable all macros:



Macro Settings

○ Disable all macros without notification

○ Disable all macros with notification

○ Disable all macros except digitally signed macros

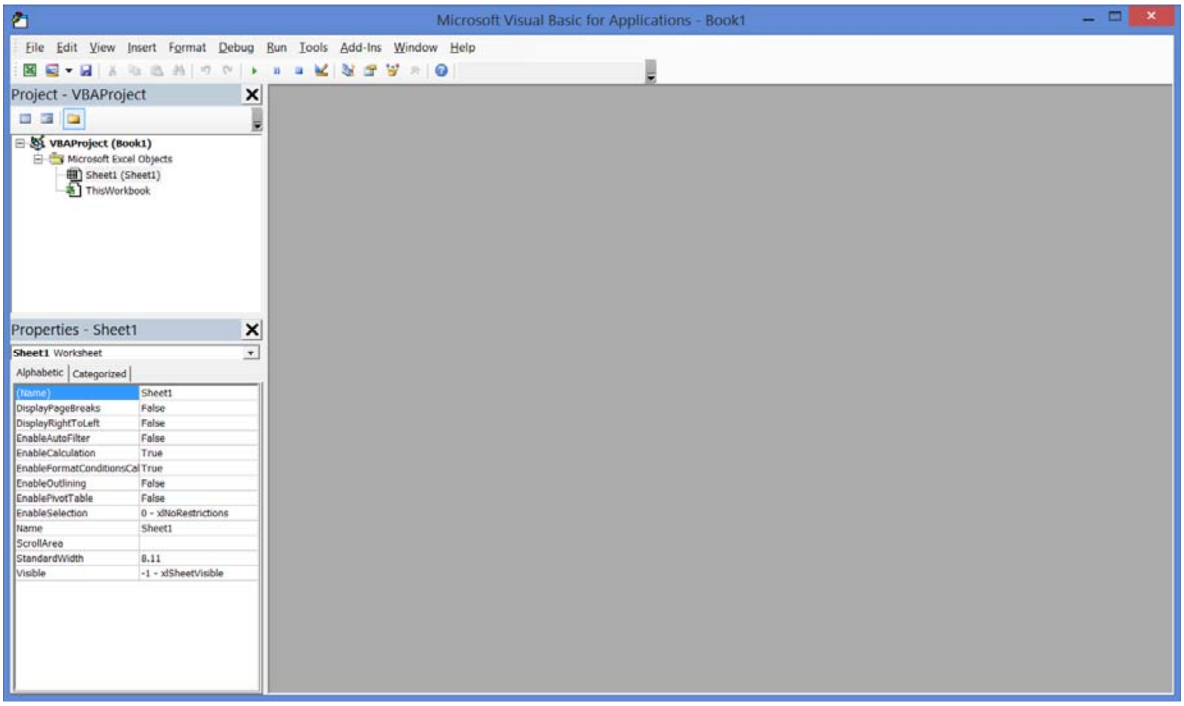
● Enable all macros (not recommended; potentially dangerous code can run)

Developer Macro Settings

□ Trust access to the VBA project object model

3. Click **OK** to close the Trust Center.

4. On the Developer tab, click Visual Basic.
   The following screen displays:

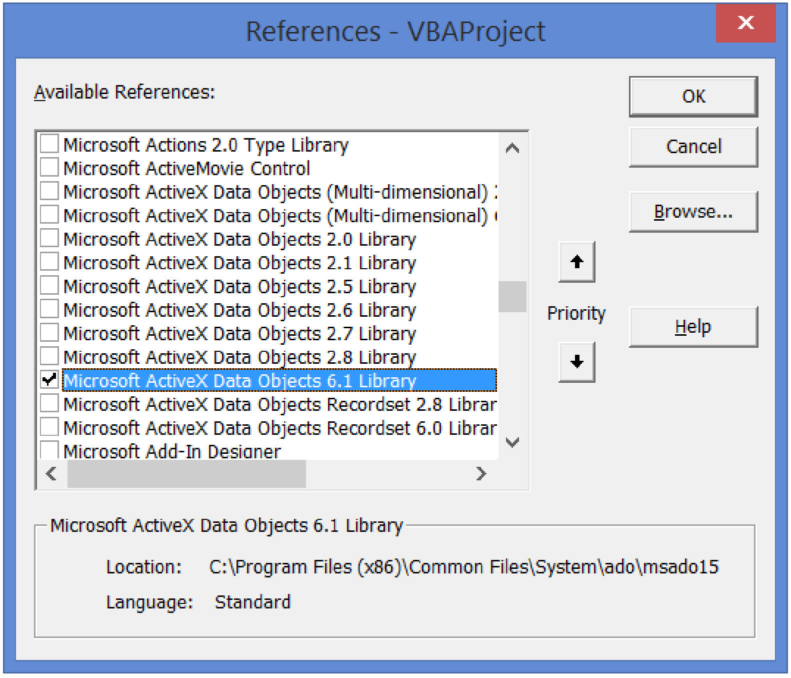


(Name)	Sheet1
DisplayPageBreaks	False
DisplayRightToLeft	False
EnableAutoFilter	False
EnableCalculation	True
EnableFormatConditionsCal	True
EnableOutlining	False
EnablePivotTable	False
EnableSelection	0 - xlNoRestrictions
Name	Sheet1
ScrollArea	
StandardWidth	8.11
Visible	-1 - xlSheetVisible

5. Ensure that the Microsoft ActiveX Data Objects Library is available:

   5.1. Select **Tools>References** in the Menu bar

   In the References – VBAProject dialog that displays, make sure that the following checkbox is checked:



- ☐ Microsoft Actions 2.0 Type Library
- ☐ Microsoft ActiveMovie Control
- ☐ Microsoft ActiveX Data Objects (Multi-dimensional) :
- ☐ Microsoft ActiveX Data Objects (Multi-dimensional) (
- ☐ Microsoft ActiveX Data Objects 2.0 Library
- ☐ Microsoft ActiveX Data Objects 2.1 Library
- ☐ Microsoft ActiveX Data Objects 2.5 Library
- ☐ Microsoft ActiveX Data Objects 2.6 Library
- ☐ Microsoft ActiveX Data Objects 2.7 Library
- ☐ Microsoft ActiveX Data Objects 2.8 Library
- ☑ Microsoft ActiveX Data Objects 6.1 Library
- ☐ Microsoft ActiveX Data Objects Recordset 2.8 Librar
- ☐ Microsoft ActiveX Data Objects Recordset 6.0 Librar
- ☐ Microsoft Add-In Designer

Microsoft ActiveX Data Objects 6.1 Library

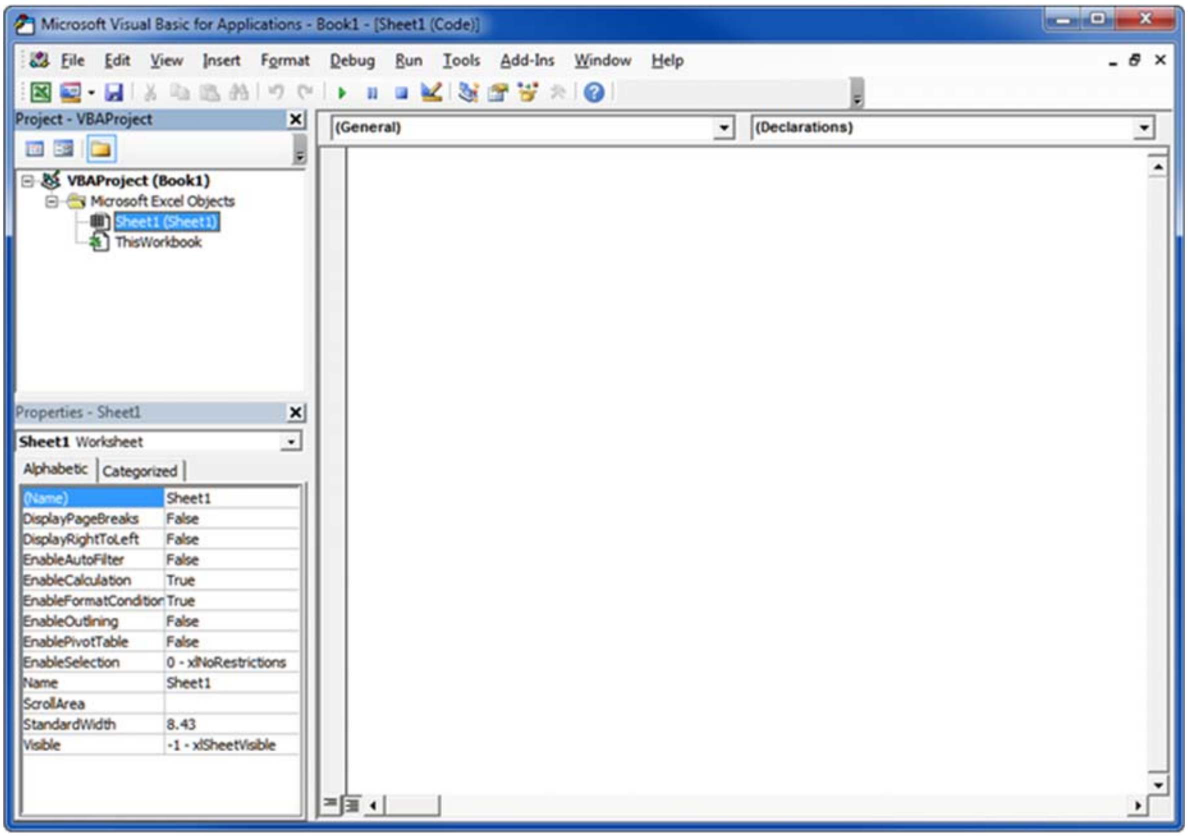
Location: C:\Program Files (x86)\Common Files\System\ado\msado15

Language: Standard

   5.2. Click **OK.**

Step 2: Design the report in Excel

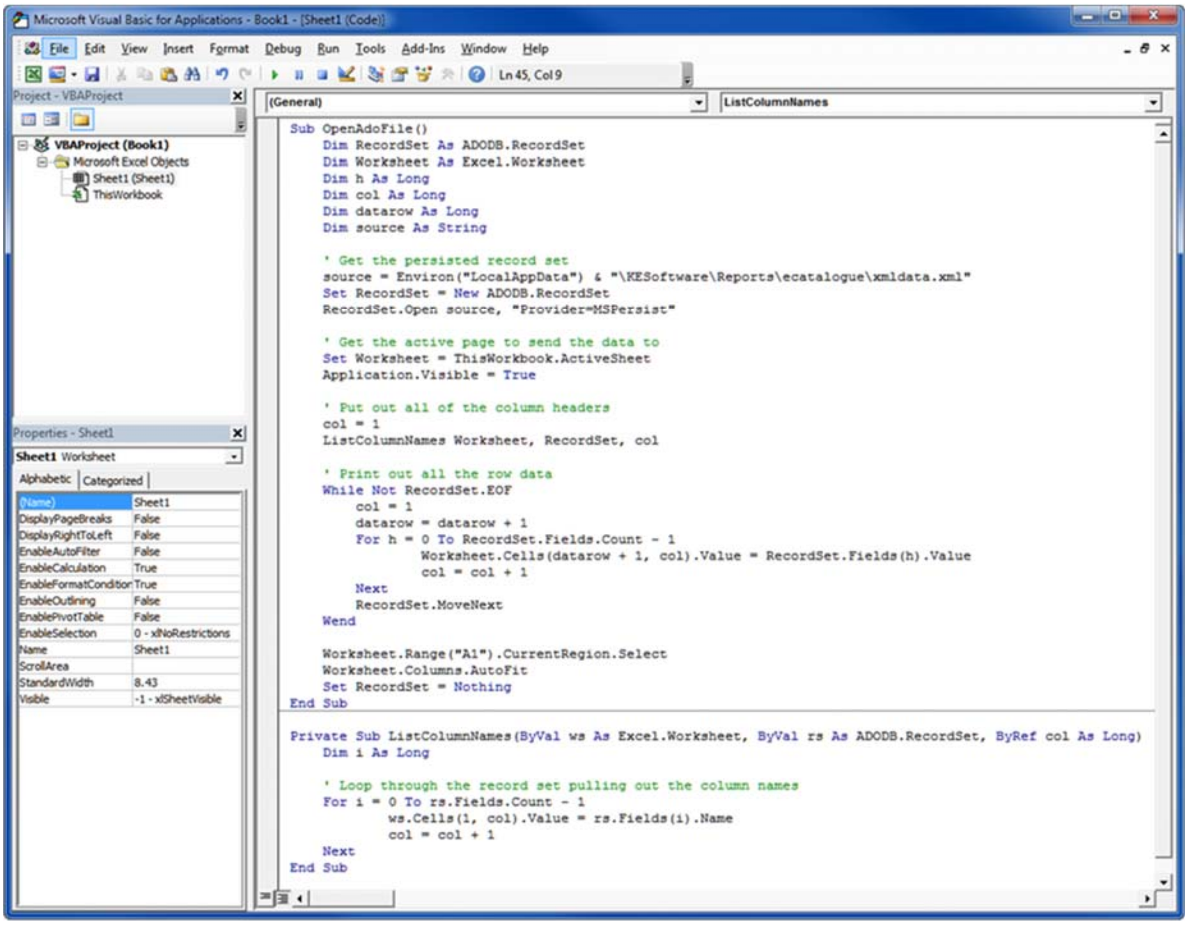
1. Double-click **Sheet1** in the VBAProject pane:



(Name)	Sheet1
DisplayPageBreaks	False
DisplayRightToLeft	False
EnableAutoFilter	False
EnableCalculation	True
EnableFormatCondition	True
EnableOutlining	False
EnablePivotTable	False
EnableSelection	0 - xlNoRestrictions
Name	Sheet1
ScrollArea	
StandardWidth	8.43
Visible	-1 - xlSheetVisible

2. Copy and paste the following VB code:

```
Sub OpenAdoFile()
    Dim RecordSet As ADODB.RecordSet
    Dim Worksheet As Excel.Worksheet
    Dim h As Long
    Dim col As Long
    Dim datarow As Long
    Dim source As String

    ' Get the persisted record set
    source = Environ("LocalAppData") & "\KESoftware\
Reports\epos\xmldata.xml"
    Set RecordSet = New ADODB.RecordSet
    RecordSet.Open source, "Provider=MSPersist"

    ' Get the active page to send the data to
    Set Worksheet = ThisWorkbook.ActiveSheet
    Application.Visible = True
    ' Put out all of the column headers
    col = 1
    ListColumnNames Worksheet, RecordSet, col

    ' Print out all the row data
    While Not RecordSet.EOF
        col = 1
        datarow = datarow + 1
        For h = 0 To RecordSet.Fields.count - 1
                Worksheet.Cells(datarow + 1, col).Value =
RecordSet.Fields(h).Value
                col = col + 1
        Next
        RecordSet.MoveNext
    Wend

    Worksheet.Range("A1").CurrentRegion.Select
    Worksheet.Columns.AutoFit
    Set RecordSet = Nothing
End Sub

Private Sub ListColumnNames(ByVal ws As Excel.Worksheet, ByVal rs
As ADODB.RecordSet, ByRef col As Long)
    Dim i As Long
    ' Loop through the record set pulling out the column names
    For i = 0 To rs.Fields.count - 1
            ws.Cells(1, col).Value = rs.Fields(i).Name
            col = col + 1
    Next
End Sub
```

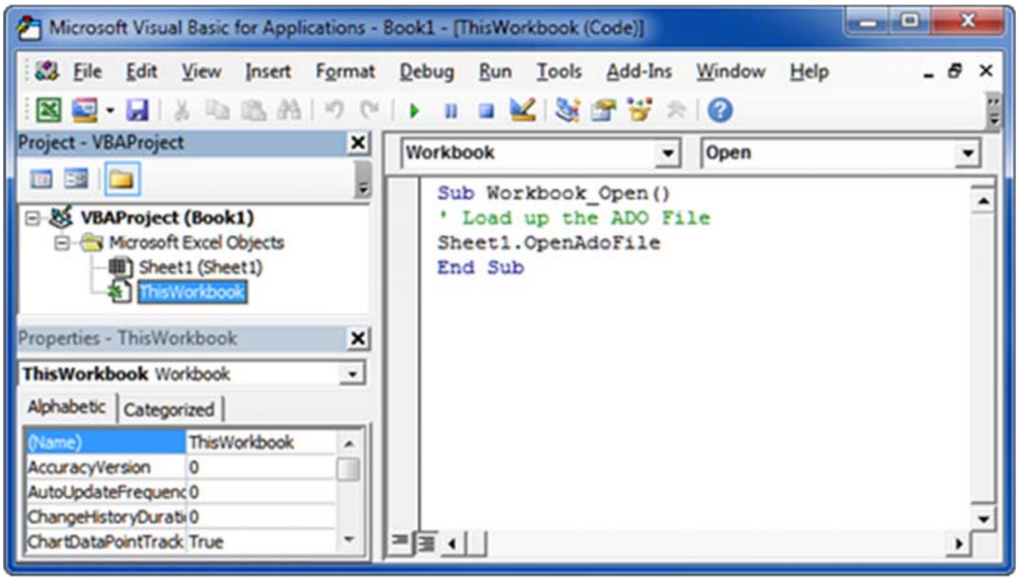


(Name)	Sheet1
DisplayPageBreaks	False
DisplayRightToLeft	False
EnableAutoFilter	False
EnableCalculation	True
EnableFormatCondition	True
EnableOutlining	False
EnablePivotTable	False
EnableSelection	0 - xlNoRestrictions
Name	Sheet1
ScrollArea	
StandardWidth	8.43
Visible	-1 - xlSheetVisible

```
Sub OpenAdoFile()
    Dim RecordSet As ADODB.RecordSet
    Dim Worksheet As Excel.Worksheet
    Dim h As Long
    Dim col As Long
    Dim datarow As Long
    Dim source As String

    ' Get the persisted record set
    source = Environ("LocalAppData") & "\KESoftware\Reports\ecatalogue\xmldata.xml"
    Set RecordSet = New ADODB.RecordSet
    RecordSet.Open source, "Provider=MSPersist"

    ' Get the active page to send the data to
    Set Worksheet = ThisWorkbook.ActiveSheet
    Application.Visible = True

    ' Put out all of the column headers
    col = 1
    ListColumnNames Worksheet, RecordSet, col

    ' Print out all the row data
    While Not RecordSet.EOF
        col = 1
        datarow = datarow + 1
        For h = 0 To RecordSet.Fields.Count - 1
                Worksheet.Cells(datarow + 1, col).Value = RecordSet.Fields(h).Value
                col = col + 1
        Next
        RecordSet.MoveNext
    Wend

    Worksheet.Range("A1").CurrentRegion.Select
    Worksheet.Columns.AutoFit
    Set RecordSet = Nothing
End Sub

Private Sub ListColumnNames(ByVal ws As Excel.Worksheet, ByVal rs As ADODB.RecordSet, ByRef col As Long)
    Dim i As Long

    ' Loop through the record set pulling out the column names
    For i = 0 To rs.Fields.Count - 1
            ws.Cells(1, col).Value = rs.Fields(i).Name
            col = col + 1
    Next
End Sub
```

3. Double-click **ThisWorbook** in the VBAProject pane and copy and paste the following code:

   ```
   Sub Workbook_Open()
   ' Load up the ADO File
   Sheet1.OpenAdoFile
   End Sub
   ```

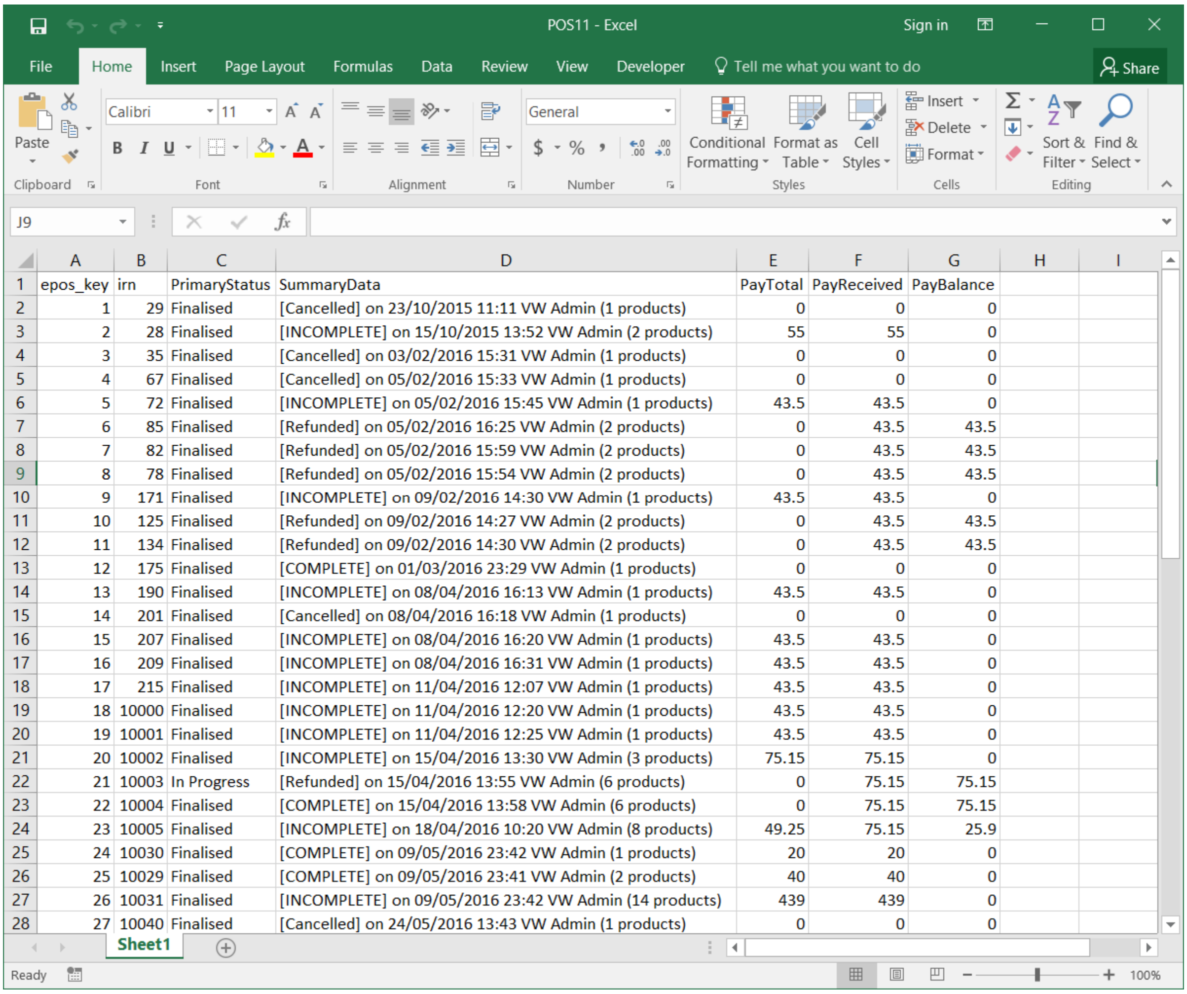

Microsoft Visual Basic for Applications - Book1 - [ThisWorkbook (Code)]
File Edit View Insert Format Debug Run Tools Add-Ins Window Help
Project - VBAProject
VBAProject (Book1)
Microsoft Excel Objects
Sheet1 (Sheet1)
ThisWorkbook
Properties - ThisWorkbook
ThisWorkbook Workbook
Alphabetic
Categorized
Workbook
Open

(Name)	ThisWorkbook
AccuracyVersion	0
AutoUpdateFrequenc	0
ChangeHistoryDurati	0
ChartDataPointTrack	True

```
Sub Workbook_Open()
' Load up the ADO File
Sheet1.OpenAdoFile
End Sub
```

4. Save the report as macro enabled and upload it to your Vitalware report (page 22) on the Report Type tab of the Report Properties box.

When the report is run in Vitalware, an Excel report is generated:

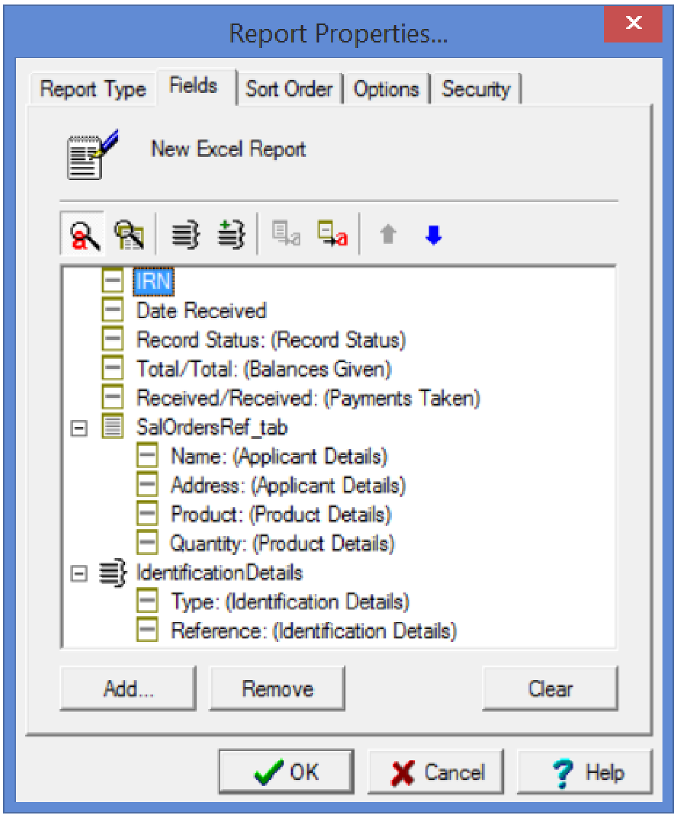


	A	B	C	D	E	F	G	H	I
1	epos_key	irn	PrimaryStatus	SummaryData	PayTotal	PayReceived	PayBalance		
2	1	29	Finalised	[Cancelled] on 23/10/2015 11:11 VW Admin (1 products)	0	0	0		
3	2	28	Finalised	[INCOMPLETE] on 15/10/2015 13:52 VW Admin (2 products)	55	55	0		
4	3	35	Finalised	[Cancelled] on 03/02/2016 15:31 VW Admin (1 products)	0	0	0		
5	4	67	Finalised	[Cancelled] on 05/02/2016 15:33 VW Admin (1 products)	0	0	0		
6	5	72	Finalised	[INCOMPLETE] on 05/02/2016 15:45 VW Admin (1 products)	43.5	43.5	0		
7	6	85	Finalised	[Refunded] on 05/02/2016 16:25 VW Admin (2 products)	0	43.5	43.5		
8	7	82	Finalised	[Refunded] on 05/02/2016 15:59 VW Admin (2 products)	0	43.5	43.5		
9	8	78	Finalised	[Refunded] on 05/02/2016 15:54 VW Admin (2 products)	0	43.5	43.5		
10	9	171	Finalised	[INCOMPLETE] on 09/02/2016 14:30 VW Admin (1 products)	43.5	43.5	0		
11	10	125	Finalised	[Refunded] on 09/02/2016 14:27 VW Admin (2 products)	0	43.5	43.5		
12	11	134	Finalised	[Refunded] on 09/02/2016 14:30 VW Admin (2 products)	0	43.5	43.5		
13	12	175	Finalised	[COMPLETE] on 01/03/2016 23:29 VW Admin (1 products)	0	0	0		
14	13	190	Finalised	[INCOMPLETE] on 08/04/2016 16:13 VW Admin (1 products)	43.5	43.5	0		
15	14	201	Finalised	[Cancelled] on 08/04/2016 16:18 VW Admin (1 products)	0	0	0		
16	15	207	Finalised	[INCOMPLETE] on 08/04/2016 16:20 VW Admin (1 products)	43.5	43.5	0		
17	16	209	Finalised	[INCOMPLETE] on 08/04/2016 16:31 VW Admin (1 products)	43.5	43.5	0		
18	17	215	Finalised	[INCOMPLETE] on 11/04/2016 12:07 VW Admin (1 products)	43.5	43.5	0		
19	18	10000	Finalised	[INCOMPLETE] on 11/04/2016 12:20 VW Admin (1 products)	43.5	43.5	0		
20	19	10001	Finalised	[INCOMPLETE] on 11/04/2016 12:25 VW Admin (1 products)	43.5	43.5	0		
21	20	10002	Finalised	[INCOMPLETE] on 15/04/2016 13:30 VW Admin (3 products)	75.15	75.15	0		
22	21	10003	In Progress	[Refunded] on 15/04/2016 13:55 VW Admin (6 products)	0	75.15	75.15		
23	22	10004	Finalised	[COMPLETE] on 15/04/2016 13:58 VW Admin (6 products)	0	75.15	75.15		
24	23	10005	Finalised	[INCOMPLETE] on 18/04/2016 10:20 VW Admin (8 products)	49.25	75.15	25.9		
25	24	10030	Finalised	[COMPLETE] on 09/05/2016 23:42 VW Admin (1 products)	20	20	0		
26	25	10029	Finalised	[COMPLETE] on 09/05/2016 23:41 VW Admin (2 products)	40	40	0		
27	26	10031	Finalised	[INCOMPLETE] on 09/05/2016 23:42 VW Admin (14 products)	439	439	0		
28	27	10040	Finalised	[Cancelled] on 24/05/2016 13:43 VW Admin (1 products)	0	0	0		

How to create an Excel Report with nested tables using the ADO RecordSet

1. Repeat Step1: Create a new report in Vitalware (page 22).

   For this example, the following fields were selected. Note the two nested tables – *SalOrdersRef_tab* and *IdentificationDetails*:



IRN
Date Received
Record Status: (Record Status)
Total/Total: (Balances Given)
Received/Received: (Payments Taken)
SalOrdersRef_tab
- Name: (Applicant Details)
- Address: (Applicant Details)
- Product: (Product Details)
- Quantity: (Product Details)

IdentificationDetails
- Type: (Identification Details)
- Reference: (Identification Details)

2. In Excel, click Visual Basic on the Developers tab.
3. Double-click **Sheet1** in the VBAProject pane:
4. Copy and paste the following VB code:

```
Sub Read_XML_Data()

    Dim rst As ADODB.Recordset
    Dim Worksheet As Excel.Worksheet
    Dim i As Long
    Dim j As Long
    Dim source As String
    Dim datarow As Long
    Dim saverow As Long
    Dim lastrow As Long
    Dim col As Long

    ' The next declaration is a little odd. Its needed in cases
where the entire value
    ' of a nested table is blank. In these cases it is necessary
to force a number of columns to be skipped when printing
    ' out field values. Oddly, as long as a nested table has at
least one value, then there is no issue.
    ' There is only a need to declare one variable for each nested
table.
    ' In this example there are only two nested tables so two
declarations are needed
    ' The value assigned to each variable will depend on the
number of fields in that nested table.
    ' In this example the first nested table is the
SalOrdersRef_tab, which has four fields, i.e. AppName,
AppAddress, OrdProduct and OrdQuantity
    ' and the second nested table, i.e IdentificationDetails, has
2 fields, i.e. IdeIdentificationType and
IdeIndentificationReference

    Dim firstnestedtable As Long
    Dim secondnestedtable As Long
    Dim nestedtablecount As Long

    firstnestedtable = 4
    secondnestedtable = 2
    nestedtablecount = 1

    ' Get the persisted record set
    source = Environ("LocalAppData") &
"\KESoftware\Reports\epos\xmldata.xml"
    Set rst = New ADODB.Recordset
    rst.Open source, "Provider=MSPersist"
```

```
' Get the active page to send the data to
Set Worksheet = ThisWorkbook.ActiveSheet
Application.Visible = True

'Add column labels
Worksheet.Cells(1, 1).Select
ActiveCell.EntireRow.Insert
Worksheet.Cells(1, 1).Value = "Record No"
Worksheet.Cells(1, 2).Value = "IRN No"
Worksheet.Cells(1, 3).Value = "Date Rec'vd"
Worksheet.Cells(1, 4).Value = "Status"
Worksheet.Cells(1, 5).Value = "Total"
Worksheet.Cells(1, 6).Value = "Paid"
Worksheet.Cells(1, 7).Value = "Applicant"
Worksheet.Cells(1, 8).Value = "Address"
Worksheet.Cells(1, 9).Value = "Product"
Worksheet.Cells(1, 10).Value = "Qty"
Worksheet.Cells(1, 11).Value = "Identification Type"
Worksheet.Cells(1, 12).Value = "Id Value"

col = 1
' Start printing data from Row 3
datarow = 3
lastrow = datarow
While Not rst.EOF
    col = 1

    If datarow < lastrow Then
        datarow = lastrow
    End If

    For j = 0 To rst.Fields.Count - 1
        If rst.Fields(j).Type = adChapter Then
            If rst.Fields(j).Value.BOF Then
                Worksheet.Cells(datarow, col).Value = ""
                If nestedtablecount = 1 Then
                    col = col + firstnestedtable
                    nestedtablecount = nestedtablecount + 1
                ElseIf nestedtablecount = 2 Then
                    col = col + secondnestedtable
                    nestedtablecount = nestedtablecount + 1
                End If
            Else
```

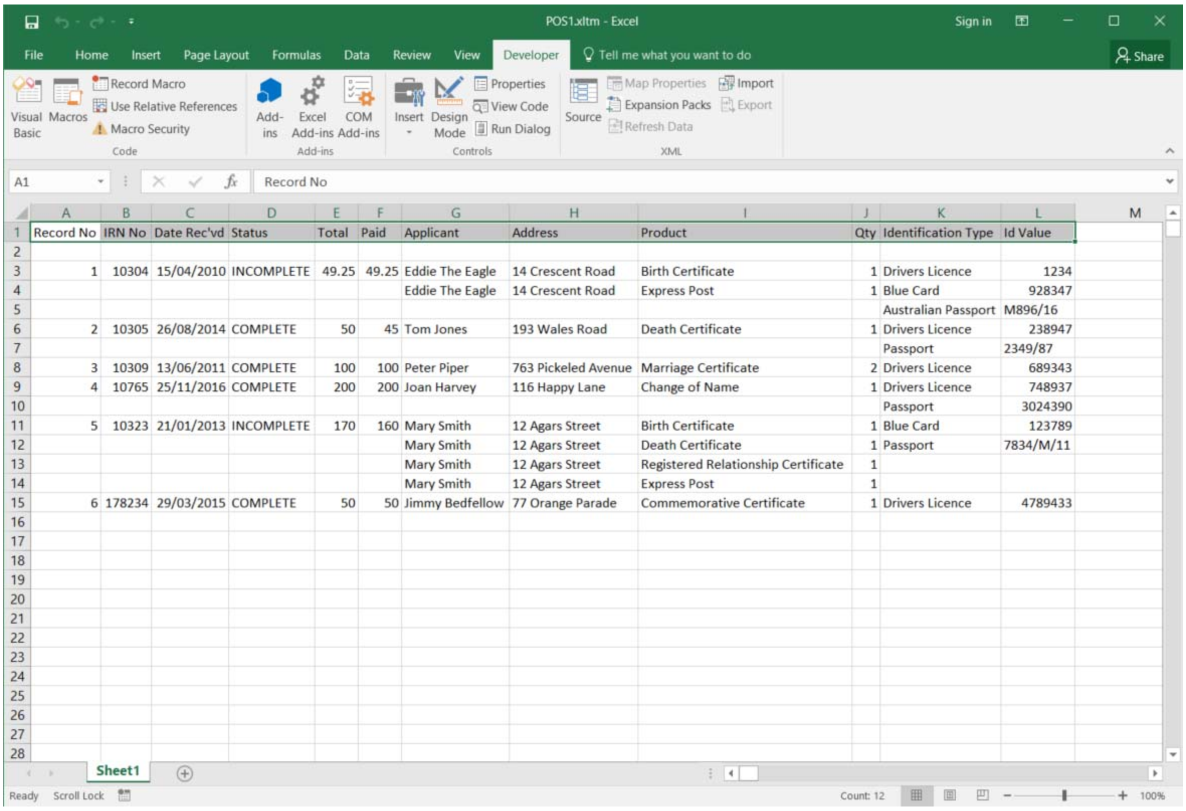
```
                    If rst.Fields(j).Value.EOF Then
                        Worksheet.Cells(datarow, col).Value = ""
                        If nestedtablecount = 1 Then
                            col = col + firstnestedtable
                            nestedtablecount = nestedtablecount
+ 1
                        ElseIf nestedtablecount = 2 Then
                            col = col + secondnestedtable
                            nestedtablecount = nestedtablecount
+ 1
                        End If
                    Else
                        saverow = datarow
                        ListNestedValues Worksheet,
rst.Fields(j).Value, col, datarow, lastrow, saverow,
nestedtablecount
                    End If
                End If
            Else
                If IsNull(rst.Fields(j).Value) Then
                    Worksheet.Cells(datarow, col).Value = ""
                Else
                    Worksheet.Cells(datarow, col).Value =
rst.Fields(j).Value
                End If
                col = col + 1
            End If
        Next
        rst.MoveNext
        datarow = datarow + 1
        nestedtablecount = 1
    Wend

    'Closing the recordset.
    rst.Close

    'Release object from memory.

    Worksheet.Range("A1").CurrentRegion.Select
    Worksheet.Columns.AutoFit
    Set rst = Nothing

End Sub
```

```
Private Sub ListNestedValues(ByVal ws As Excel.Worksheet, ByVal
rs As ADODB.Recordset, ByRef col As Long, ByRef datarow As Long,
ByRef lastrow As Long, ByRef saverow As Long, ByRef
nestedtablecount As Long)
    Dim i As Long
    Dim j As Long
    Dim startrow As Long

    ' Loop through a nested table pulling out the row values
    j = 0
    startrow = saverow
    While Not rs.EOF
        Max = 1
        j = col
        For i = 0 To rs.Fields.Count - 1
            ' Don't print key values
            If rs.Fields(i).Name <> "SalOrdersRef_key" And
rs.Fields(i).Name <> "IdentificationDetails_key" And
rs.Fields(i).Name <> "epos_key" _
            Then
                If IsNull(rs.Fields(i).Value) Then
                    ws.Cells(startrow + 1, j).Value = ""
                    j = j + 1
                Else
                    If rs.Fields(i).Type = adChapter Then
                        ListNestedValues ws, rs.Fields(i).Value,
j, datarow, lastrow, saverow, nestedtablecount
                        datarow = startrow
                    Else
                        ws.Cells(startrow, j).Value =
rs.Fields(i).Value
                        j = j + 1
                    End If
                End If
            End If
        Next
        rs.MoveNext
        startrow = startrow + 1
    Wend

    If (j > 0) Then
        col = j
    End If
```

```
        If startrow > lastrow Then
            lastrow = startrow
        End If

        nestedtablecount = nestedtablecount + 1
       End Sub
```

5. Double-click **ThisWorbook** in the VBAProject pane and copy and paste the following code:
   ```
   Sub Workbook_Open()
   ' Load up the ADO File
   Sheet1.Read_XML_Data
   End Sub
   ```
6. Save the report and upload it to your Vitalware report (page 22) on the Report Type tab of the Report Properties box.

When the report is run in Vitalware, an Excel report is generated:



Record No	IRN No	Date Rec'vd	Status	Total	Paid	Applicant	Address	Product	Qty	Identification Type	Id Value
1	10304	15/04/2010	INCOMPLETE	49.25	49.25	Eddie The Eagle	14 Crescent Road	Birth Certificate	1	Drivers Licence	1234
						Eddie The Eagle	14 Crescent Road	Express Post	1	Blue Card	928347
										Australian Passport	M896/16
2	10305	26/08/2014	COMPLETE	50	45	Tom Jones	193 Wales Road	Death Certificate	1	Drivers Licence	238947
										Passport	2349/87
3	10309	13/06/2011	COMPLETE	100	100	Peter Piper	763 Pickeled Avenue	Marriage Certificate	2	Drivers Licence	689343
4	10765	25/11/2016	COMPLETE	200	200	Joan Harvey	116 Happy Lane	Change of Name	1	Drivers Licence	748937
										Passport	3024390
5	10323	21/01/2013	INCOMPLETE	170	160	Mary Smith	12 Agars Street	Birth Certificate	1	Blue Card	123789
						Mary Smith	12 Agars Street	Death Certificate	1	Passport	7834/M/11
						Mary Smith	12 Agars Street	Registered Relationship Certificate	1		
						Mary Smith	12 Agars Street	Express Post	1		
6	178234	29/03/2015	COMPLETE	50	50	Jimmy Bedfellow	77 Orange Parade	Commemorative Certificate	1	Drivers Licence	4789433

SECTION 4

Registry entries

The Type Registry entry indicates which export type to use for each report request.

The format of this Registry entry is;

`System|Setting|Reports|Type|Crystal CSV|`*value*

value is `0` or `1`:

`0` Generates data in the existing format.

`1` Generates data in the new Crystal ODBC format.

If this entry is not present, a *value* of `0` is assumed.

`System|Setting|Reports|Type|Crystal ADO|`*value*

value is `0` or `2`:

`0` Generates data in the existing format.

`2` Generates data in the new Crystal ADO record set.

If this entry is not present, a *value* of `0` is assumed.

`System|Setting|Reports|Type|Microsoft ADO|`*value*

where:

value is `0` or `3`:

`0` Generates data in the existing format.

`3` Generates data in the new Microsoft ADO format.

If this entry is not present, a *value* of `0` is assumed.

Index